Space Object Identification Using Feature Space Trajectory Neural Networks

THESIS

Neal W. Bruegger
Captain, USAF

AFIT/GOR/ENG/97M-02

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

19970429 227

DTIC QUALITY INSPECTED 1

AFIT/GOR/ENG/97M-02

Space Object Identification Using

Feature Space Trajectory Neural Networks

THESIS

Neal W. Bruegger
Captain, USAF

AFIT/GOR/ENG/97M-02

Approved for public release; distribution unlimited

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government.

AFIT/GOR/ENG/97M-02

Space Object Identification Using Feature Space Trajectory Neural Networks

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

Neal W. Bruegger, B.S.

Captain, USAF

March, 1997

Approved for public release; distribution unlimited

THESIS APPROVAL

Student: Neal W. Bruegger, Capt, USAF   Class: GOR-97M

Title: Space Object Identification Using Feature Space Trajectory Neural Networks

Defense Date: 20 February 1997

Committee:	Name/Title/Department	Signature
Advisor	Steven K. Rogers Professor Department of Electrical Engineering	
Reader	Martin P. DeSimio Assistant Professor Department of Electrical Engineering	Martin P. DeSimio
Reader	Kenneth W. Bauer Professor Department of Operational Sciences	Kenneth W. Bauer

Acknowledgements

I would like to thank my advisor, Dr. Steve Rogers, for guiding my work to completion. His advice kept me on-track and allowed me to finish without much of the stress and turbulence often experienced by my classmates. His classes also made my AFIT experience far more enjoyable than it could have been otherwise.

I would also like to thank Dr. Marty DeSimio and Dr. Ken Bauer for taking an interest in my work and staying so involved in its progress. Their help aided immeasurably in its completion. In addition, Dr. Byron Welsh and Major Rob Brigantic were instrumental in educating me on the concepts of turbulence and the use of HYSIM.

I am also compelled to acknowledge my classmates for the support and comradery they showed to myself and each other. They truly preserved my sanity at times.

Finally, I would like to thank my wife, Jennifer, for willingly becoming an "AFIT Widow" for 18 months and giving me all the love, support, and friendship I could have ever imagined. I love you.

Neal W. Bruegger

Table of Contents

Page

List of Figures

Figure		Page

List of Tables

AFIT/GOR/ENG/97M-02

Abstract

The Feature Space Trajectory Neural Network (FSTNN) is a simple yet powerful pattern recognition tool developed by Neiberg and Casasent for use in an Automatic Target Recognition System. In the FSTNN algorithm, trajectories through the feature space representing pose transitions of objects are generated. An unknown object can be classified according to the nearest trajectory of a known object. Furthermore, the pose of the unknown object can be approximated according to the position of the closest point on the closest trajectory.

Since the FSTNN was developed, it has been used on various problems including speaker identification and space object identification. However, in these types of problems, the test set represents time-series data rather than an independent set of points. Since the distance metric of the standard FSTNN treats each test point independently without regard to its position in the sequence, the FSTNN can yield less than optimal results in these problems.

Two methods for incorporating sequence information into the FSTNN algorithm are presented. These methods, Dynamic Time Warping (DTW) and Uniform Time Warping (UTW), are described and compared to the standard FSTNN performance on the space object identification problem. Both reduce error induced by improper synchronization of the test and training sequences and make the FSTNN more generally applicable to a wide variety of pattern recognition problems. They incorporate sequencing information by synchronizing the test and training trajectories. DTW accomplishes this "on-the-fly" as the sequence progresses while UTW uniformly compensates for temporal differences across the trajectories. These algorithms improve the maximum probability of false alarm (P_{FA}) of the standard FSTNN by an average of 10.18% and 27.69%, respectively, although UTW is less consistent in its results.

Space Object Identification Using

Feature Space Trajectory Neural Networks

I. Introduction

1.1 Space Object Identification

Space Object Identification (SOI) is the process of producing images of man-made satellites from a ground station such as the Air Force Maui Optical Station (AMOS) and analyzing those images in order to determine several characteristics of the satellite: (1) Location, in terms of orbital dynamics; (2) Behavior, such as spin or wobble; (3) Identification; and (4) Capability [21]. This task is currently performed by analysts. The analysis is therefore dependent upon the analysts' training and experience and is naturally very time consuming. A computer-based tool which could perform this analysis, or at least accomplish some initial screening of the data would be extremely helpful in reducing analysis time and, perhaps, human error. Feature Space Trajectory Neural Networks (FSTNNs) are a type of pattern recognition algorithm which can be used in solving this problem [4, 6].

1.2 Previous Work

Neiberg and Casasent [9, 12, 13, 14, 10, 11] developed the FSTNN, an extremely useful tool in many pattern recognition problems where data is classified according to an approximated path through the feature space. Several major advantages are realized in using the FSTNN as opposed to one of many various forms of the standard neural network (e.g. backprop multilayer perceptron (MLP) [16]). One of these is the fact that an FSTNN can successfully classify overlapping data sets because its decision rule is based upon distance to a class trajectory, rather than spatial distributions of the data in the feature space. The absence of parameters, such as momentum and learning rate in an MLP, which must be heuristically

set and whose optimum values are extremely data-dependent, is another advantage of the FSTNN. In addition to classification, the FSTNN is useful for pose estimation because of the information contained in the spatial position of each point on the trajectory. For example, if each vertex on the trajectory represents an image of an object at a particular aspect view, then the trajectory represents the continuous range of aspect views. Thus, the aspect of any point on the trajectory between two vertices can be interpolated from the known aspect of the vertices.

Neiberg and Casasent tested their algorithm on a simulated infrared military vehicle database containing 126 objects. They compared the FSTNN to 3-layer backpropagation network (BPNN), piecewise quadratic NN (PQNN), and direct-storage nearest neighbor (DSNN) classifiers. The FSTNN achieved 100% classification, while the others achieved only 91.7%, 93.3%, and 93.3%, respectively. In addition, the FSTNN is a much simpler algorithm which does not require extensive training, whereas the other methods may require thousands of iterations to achieve their highest classification success rates. The aspect estimation obtained from the FSTNN was accurate to 5.7°. This information is not available from the other classifiers.

In 1995, Brandstrom compared the use of Feature Space Trajectory Neural Nets (FSTNN) and Hidden Markov Models (HMM) to determine a satellite's behavior [4, 6]. As Brandstrom points out, the satellite's location and identification are usually known by its orbital parameters before the images are obtained. Therefore, the scope of his thesis was limited to the problem of determining a satellite's behavior. In particular, the goal was to classify the behavior of a satellite as either normal or anomalous. He concluded that the FSTNN solution was superior to the HMM solution.

The FSTNN was used by Neiberg and Casasent for identification and pose estimation of single object instances. However, the FSTNN has more general uses in comparing not only single test points to a class trajectory, but also test data which are themselves trajectories. This is the case with the SOI problem; sequences of images form a test trajectory which must be compared to one or more class trajectories to determine the closest match. Unfortunately, when trajectory-to-trajectory comparisons are made, discriminating information is contained

not only in the location of the data points in the feature space, but also in the sequence those points are presented, especially when the system being modeled is a real-time event sequence. Neiberg and Casasent's original algorithm does not associate any sequence or time information with the test data. In 1996, they published a paper in which they present three methods for trajectory-to-trajectory comparisons used in the context of Automatic Target Recognition (ATR) [13]. These methods, however, are not ideal for use in the SOI problem, as explained in Section 2.2.4.

Incorporating sequence information is merely a problem of deciding what portion of the class trajectory is acceptable for a given test point to project to. This can be accomplished in one of two ways: dynamically assigning acceptable portions based on projection locations of previous test points (dynamic time warping) or uniformly assigning projection points based upon corresponding distances along the trajectories (uniform time warping).

Brandstrom developed a modification to the algorithm in which he obtained a sequence metric in addition to the distance measure. This thesis is a follow-up effort to his work which will focus on improving the FSTNN to better incorporate sequence information. While Brandstrom achieved excellent results, modified FSTNNs have been developed using Dynamic Time Warping and Uniform Time Warping that have the advantage of being more generally applicable to a wider range of pattern recognition problems. The results of this thesis will be a significant contribution to the field of pattern recognition by developing a spatio-temporal FSTNN (as opposed to the spatial FSTNN developed by Neiberg and Casasent).

1.3 Problem Statement

Improve the Feature Space Trajectory Neural Network by incorporating sequencing information into the algorithm.

1.4 Scope

1.4.1 Simulated Data. The data to be processed was generated by Brandstrom using the SatTools software package. It is assumed that these images are adequate substitutes for

the actual images obtained by AMOS. Since most of the actual images produced at AMOS are classified, the use of the unclassified simulated images is necessary to allow public distribution of this thesis.

1.4.2 Feature Selection. The results of any pattern recognition process are only as good as the features selected. Brandstrom conducted an analysis of various features, and was able to produce good results for several types of features. Since this thesis is concerned only with the efficiency of the FSTNN itself, a single set of features for which Brandstrom achieved good results will be used. The 8-wedge Fourier features, as well as the image processing used to obtain them, are assumed to be adequate for the techniques presented in this thesis. A more complete description of these features and why they were selected is included in Section 3.2.

1.4.3 Continuous Feature State Transitions. The assumption that the trajectories represent continuous feature state transitions is fundamental to FSTNN methodology. While the FSTNN has been shown to be useful in pattern recognition problems, this assumption has never been rigorously proven. Since Brandstrom has already demonstrated the usefulness of the FSTNN in the SOI problem, the experiments in this thesis will presume upon the legitimacy of this assumption as well as the adequacy of a linear representation of these transitions.

1.4.4 FSTNN Scope. A comparison of three modified FSTNN algorithms applied to the SOI problem will be accomplished:

1) Standard FSTNN
2) FSTNN with Dynamic Time Warping
3) FSTNN with Uniform Time Warping

In addition, an investigation of the saliency of the individual features will be accomplished. To date, no standard metrics for testing feature saliency in an FSTNN have been published.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. The second chapter provides a basic overview of pattern recognition concepts as well as a conceptual overview of Neiberg and Casasent's FSTNN and the three modified forms to be tested. Chapter III discusses the methodology of the analysis, including a brief discussion of the simulated data, feature selection, and the actual implementation of the three modified FSTNN algorithms. Chapter IV contains the experiment results and analysis. The last chapter presents the conclusions of this research along with ideas for future research related to this topic.

II. Background

2.1 Classical Pattern Recognition

Pattern Recognition is simply the ability to discriminate between several classes of objects or events. In order to do this, features must be selected which describe the objects in sufficient detail to discriminate between different classes.

Classical pattern recognition typically involves calculating distributions for each class of data so that any point in the feature space will have a greater *a posterior* probability of being in one class than the others. A transition between regions in the feature space corresponding to different classes is called a "decision boundary". In addition to classical statistical methods which assume a distributional form for the data (i.e. Gaussian) and estimate its parameters, many nonparametric neural network algorithms, such as the Multilayer Perceptron and its variations [8] exist which iteratively derive the distributions of the test data or are trained to classify the data directly.

2.2 Feature Space Trajectory Neural Networks

The FSTNN is similar to a nearest neighbor classifier, but exploits the assumption that members of the same class will follow a continuous trajectory through the feature space. Because of this assumption of form, the FSTNN is a parametric approximator. For example, suppose an aircraft is photographed at several different aspects. The appropriate features are extracted and each image is plotted as a data point in the feature space. A line linking adjacent aspect images can be interpolated, representing the continuous range of aspects between those of the two endpoint images. This interpolation allows for good generalization and correct classification of points completely outside the training set.

2.2.1 Structure. In the FSTNN, the input nodes N_1 represent features and the mid-layer nodes N_2 represent the links along the trajectory, as shown in Figure 1. In this figure, P_1 represents a set of unknown test points. These points are tested one at a time. That

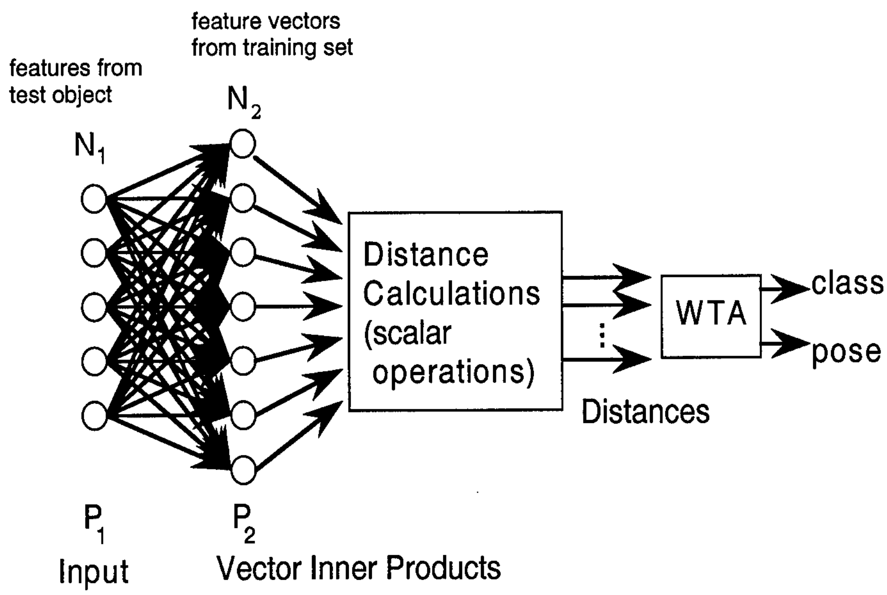


Figure 1. FSTNN NN architecture. [12]

is, the nodes of N_1 represent the features of a test point, p, not the points in the set P_1. A single unknown test data point, p, is entered into the input layer, and its Vector Inner Product with each of the links in the training trajectory is calculated to find the set of closest points, P_2, on each link to p. The distances to each link are then computed, and the closest link wins, with its corresponding projection point, p_2, representing the pose of p interpolated from the known poses of the link endpoints.

If several trajectories exist, each representing a different class of data, a test point can be classified according to the closest link out of all trajectories. In this manner, the FSTNN can be used for classification and pose estimation. In addition to identifying class by the closest trajectory, threshold distances can be added to identify points which do not belong to any class. Appropriate thresholds with respect to the SOI problem are discussed in Section 3.4.

2.2.2 Problem of Incorporating Sequence Information. While the FSTNN is useful for class determination and pose estimation, it can also test the class of a sequence of data points. In the SOI problem, a satellite is photographed at constant time intervals as

it passes over an observatory on Maui. Since the pose of the satellite steadily changes with respect to the observatory, the feature space trajectory formed from the sequence of satellite images reflects this continuous change in pose. This trajectory can be compared to known trajectories to determine the behavior of its orbit (normal/anomalous) [4, 6]. However, the FSTNN algorithm does not encode any sequence information into the set of test data points. It simply compares the test set as an independent set of feature vectors rather than a sequence of vectors. Since the FSTNN measures the distance of each point to its closest link on the trained trajectory, inaccurate classification can result when the trajectories intersect themselves or otherwise have links very near to other links on the same trajectories.

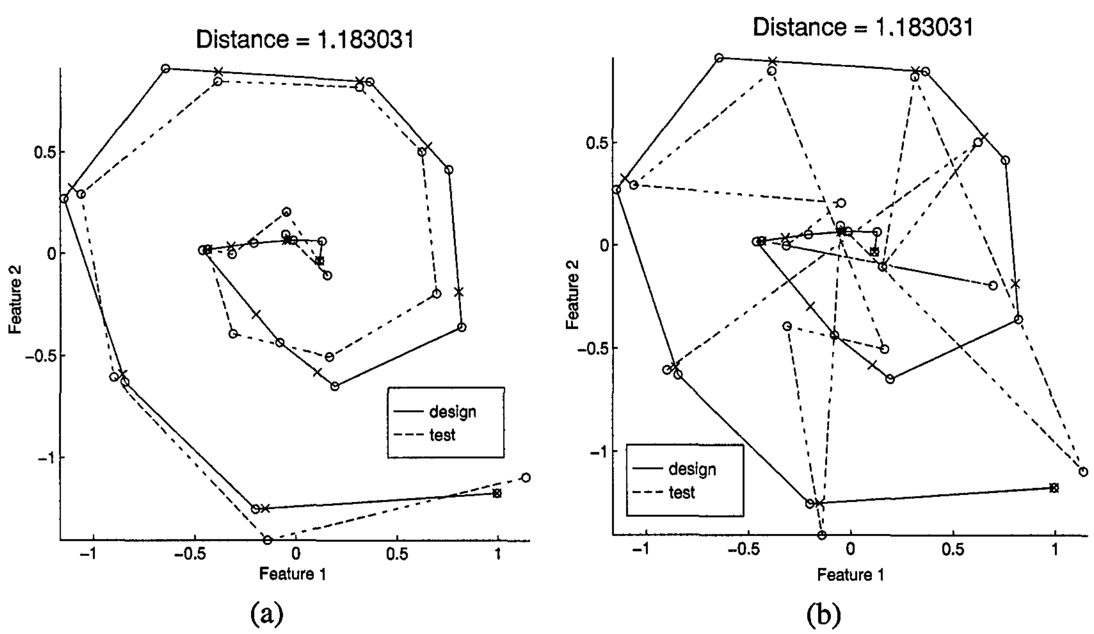


Figure 2. Comparison of (a) normal test spiral and (b) "scrambled" test spiral to design spiral. Distances are equal, even though the trajectories are different.

Figure 2 demonstrates this loss of information when the temporal structure is ignored. This example involves two sets of two-dimensional feature vectors, a test set and a training set. Graph (a) shows both sets plotted in the proper sequence, and the FSTNN distance is calculated. Because the two trajectories are consistently close from beginning to end, the FSTNN distance is an accurate reflection of how close the test trajectory is to the training

trajectory. However, if the sequence of the test trajectory was different, as in (b), a problem arises. The test trajectory is dramatically different than in (a), even though it contains the same feature vector vertices. Since the FSTNN calculates the distance of each point in the test set independently, it will arrive at the same distance measure as the test set in (a). In this case, the loss of sequence information leads to an inaccurate result.

2.2.3 Brandstrom's Solution. In addition to the distance metric, Brandstrom devised a sequence metric to indicate large deviations from expected sequencing. Each point on the test sequence was labeled 1 through n (n = 20 for his data set) from start to finish. Each link in the training, or class, trajectory was also labeled similarly. Since the test sequences and class trajectories contained the same number of points, the first point in the test set was expected to project to the first link in the class trajectory if the point indeed belonged to that class. The second point was expected to project to the second link, and so on. For each point, the difference between the expected and actual link it projected to was calculated. The sequence metric Brandstrom used was the standard deviation of these differences. Therefore, how well a test set fits a certain class depends upon how far the points are from the class trajectory (distance metric) as well as how well they matched in the proper sequence (sequence metric).

While Brandstrom's solution yielded excellent results for the SOI problem, it has several limitations which restrict its general use. The most obvious limitation is that the number of points in the test sets and the class trajectories must be equal for the sequence metric to be obtained. In addition, this method requires the interpretation of two metrics, the relative weight given to each being a subjective decision of the experimenter. Brandstrom used the distance test to find the closest match. Then the sequence test would be performed on the winner. If the sequence metric was below a certain threshold, the trajectory was considered normal. If not, it was anomalous. For the 8-wedge Fourier features, this method actually made Brandstrom's classification results worse. This is possibly due to the fact that the sequence test was applied only to the winner. If the sequence test was failed, the test sequence was labeled "anomalous" without checking to see if another normal class trajectory was close enough to pass the sequence test.

2.2.4 Neiberg and Casasent's Solutions. Neiberg and Casasent developed the FSTNN for use with an ATR system. The algorithm as originally published considered comparing individual points (objects) to the class trajectories [9, 12, 10, 13]. A subsequent publication included three methods for testing sequences of points to class trajectories [14].

2.2.4.1 Voting Method. In the voting method, each point in the test sequence is assigned to the closest class trajectory. When all points have been classified, the class with the most "votes" wins.

2.2.4.2 Vertex Sequence Method. Instead of finding the closest trajectory for each test point, the vertex sequence method calculates the distance from each point (vertex) to each class trajectory, and the distance measure of the test set to a class is defined as the average distance of the test points to that trajectory. The test sequence is then classified according to the class with the lowest average distance score. This method is quite similar to the Brandstrom distance metric without the sequence measure.

2.2.4.3 Input Trajectory Method. This method is similar to the vertex sequence method with the class trajectories and test sequence reversed. Line segments are formed between test points so that a trajectory is formed from the test data. This trajectory then serves as the training trajectory and is then classified according to the smallest average distance of the known trajectories to itself, computed according to the vertex sequence method.

2.2.4.4 Problems. While Neiberg and Casasent's methods manage to consolidate individual distance measurements into a single measure for the test set, they do not address the sequencing problem; a point's position in the sequence should affect the way its individual distance measurement to a trajectory is calculated. Therefore, these methods do not include temporal information and may lead to less accurate results depending on the form of the class trajectories.

2.2.5 *New Solutions.* Several algorithms have been developed for this thesis in an attempt to solve the sequencing problem and thereby provide either better results or better generalization of the FSTNN, or both.

2.2.5.1 *Dynamic Time Warping (DTW).* The Dynamic Time Warping algorithm used in the FSTNN evolved out of its use with nearest-neighbor classifiers [5]. In a nearest-neighbor classifier, a pre-determined number of codebook vectors are selected, either randomly or based upon some knowledge of the data being tested. The codebook vectors are then iteratively repositioned so as to minimize the distance of their closest neighboring data points. Figure 3 shows a two-dimensional example of this type of "vector quantization". Once the final position of the codebook vectors is determined, any test point is classified according to its nearest codebook vector.

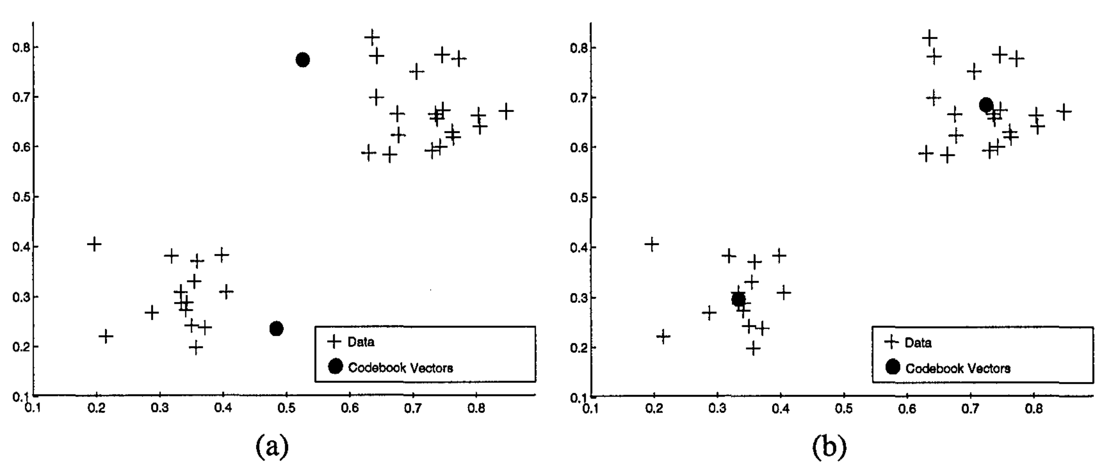


Figure 3. Example of nearest-neighbor classifier. (a) Initial position of codebook vectors (b) Final position of codebook vectors.

In speech problems [2], an utterance is segmented into frames, and features are extracted from each frame. These nodes are then classified according to the closest codebook vectors, which represent "known" utterance frames. Unfortunately, when the classified codebook vectors are placed together in sequence, they may or may not form a known or intelligible utterance. The reason for this is that the nearest-neighbor algorithm classifies each frame without regard to its order in the sequence.

Ney [15] developed a Dynamic Time Warping (DTW) algorithm for use in speech recognition to overcome this problem. What DTW essentially does is compare test frame sequences to training frame sequences. The nearest-neighbor distance calculations remain unchanged, but restrictions are placed on which codebook vectors are allowable for comparison, according to their expected order in the known utterance. Figure 4 shows a DTW distance matrix used to represent this comparison.

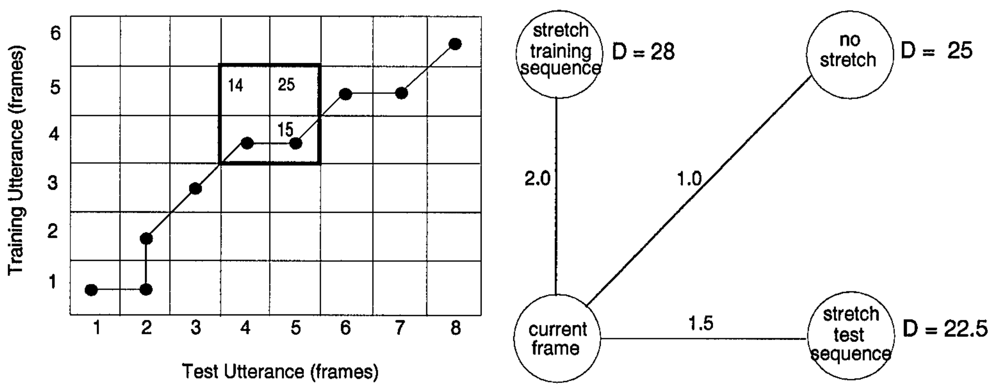


Figure 4. (a) Example of a DTW Distance Matrix. (b) Node distances are weighted by stretch factors.

The algorithm starts by measuring the distance from the first test frame to the first frame in the training utterance. Then the DTW algorithm moves from the lower left of the matrix to the upper right by comparing each test point to the adjacent frames in the training utterance. Moving horizontally implies a "time stretch" of the test trajectory with respect to the training trajectory, while moving vertically implies a time compression of the test trajectory. The shortest distance wins, but preference is given to the diagonal moves which represent an equal time progression of the test and training sequences. This is accomplished by multiplying the distance to the adjacent frames by a "stretch factor". Figure 4(b) shows the stretch factors and distances associated with moving from the fourth test frame. Even though the vertical move has the smallest distance, the stretch factors associated with these directions cause the horizontal move to be selected. DTW ensures that no test frame can be associated with a training frame which occurs more previous in time to those of the previous test frames.

With a few modifications, the DTW algorithm can be applied to the FSTNN in order to encode sequence information. This is done by modifying the DTW distance matrix to measure between test nodes and design links, rather than strict node-to-node comparisons (Figure 5). In addition, vertical moves are replaced by moving over one frame and up two. In speech problems, the objective is to map given test utterance frames to the known utterance frames so that the result is a sequence of frames which form a known utterance. The DTW algorithm compensates for non-uniform speech rates. Therefore, it is legitimate to map several codebook vectors to the same test utterance (i.e. vertical moves). However, since the FSTNN is measuring distances from the test points to the training trajectory, each test point can have only one distance (and therefore one trajectory link) associated with it. The test sequence must always progress in time, but moving ahead two links has the desired stretch effect.

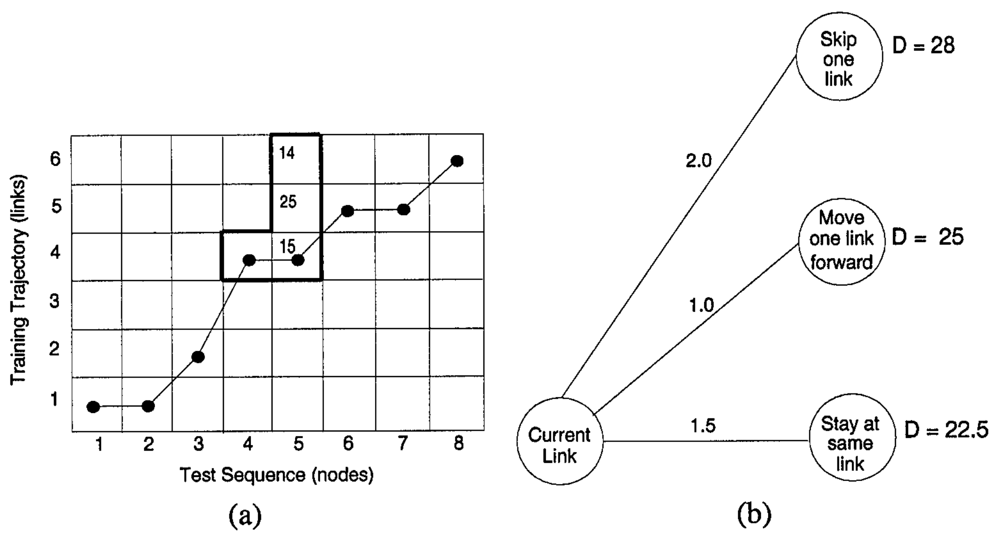
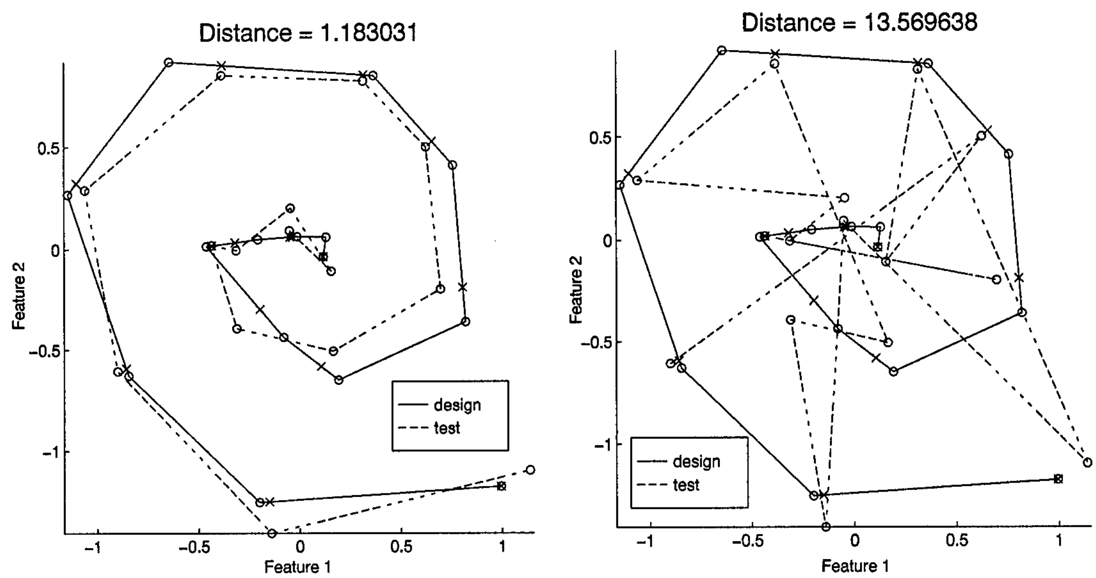


Figure 5. (a) Example of a DTW Distance Matrix for an FSTNN. (b) Node distances are weighted by stretch factors.

Using DTW, a test point can never project to a link more previous in sequence than the link selected by the previous test point. Ideally, each subsequent test point will project to sequential links in the design trajectory. As mentioned above, a point may skip ahead two links instead of one, but distance calculations are penalized by multiplying the distances to non-preferred links by the associated stretch factors.

The DTW algorithm is an excellent way to incorporate sequence information without increasing the dimensionality of the network because the sequence information is encoded in the algorithm itself rather than additional features. Recall the illustration of the sequencing problem in Figure 2. Figure 6 shows the same "scrambled" spiral sequence as in Figure 2(b), except that the comparison to the training trajectory has been accomplished using the DTW algorithm. The distance is now much higher than that of the original FSTNN comparison, even though the test vertices are the same. The DTW algorithm was successful at discriminating between the two test spirals in (a) and (b) because of the incorporation of sequence information into its algorithm. The standard FSTNN algorithm did not account for sequence and failed to discriminate between the two.



(a) (b)

Figure 6. Comparison of spiral data using (a) original FSTNN algorithm and (b) FSTNN with DTW. Distance is increased.

2.2.5.2 Uniform Time Warping. The Uniform Time Warping (UTW) algorithm is based on the assumption that the continuous feature state transition assumption which allows us to interpolate the design trajectory also applies to the test class. Therefore, if each trajectory is segmented into N equidistant links along the trajectory, the corresponding

link endpoints on each trajectory provide the endpoints of the distance calculations. This automatically synchronizes points along the trajectories in order to incorporate both sequence information and time warping (Figure 7).

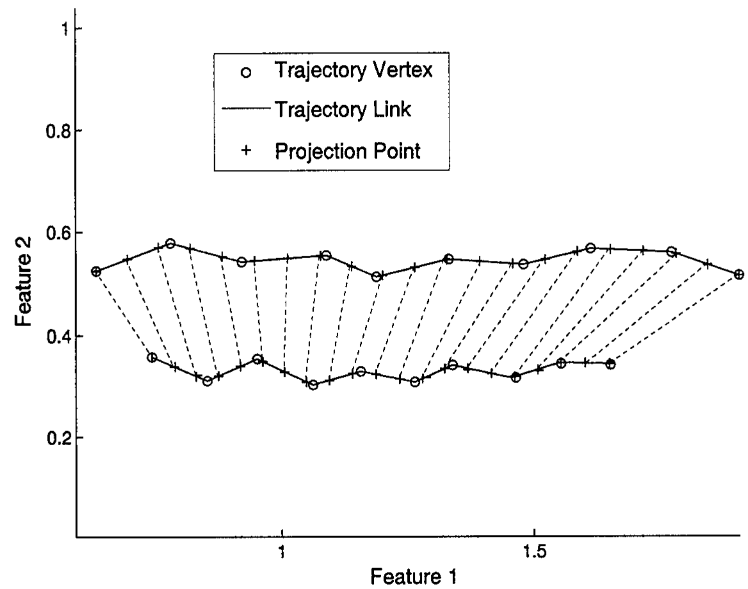


Figure 7. Comparison of two trajectories using UTW.

Figure 7 shows a two-dimensional example of UTW. The "o"s represent the original data points on each trajectory. In this case, the trajectories are each divided into 20 segments of equal length. The "+"s represent the endpoints of these links. The average distance between corresponding "+"s on each trajectory is then computed as the distance metric.

2.2.5.3 Considerations. While the UTW and DTW methods are expected to outperform the standard FSTNN algorithm, the choice of which method to use requires some insight into the nature of the data being classified. The UTW algorithm may be appropriate for image data where the image sequences are snapshots of a real-time event, such as a satellite passing overhead, and very little time variation (or at least uniform variation) occurs between instances of the real-time event. In such cases one can expect variations in trajectory lengths to be uniformly distributed along the trajectory because of the common time reference of the trajectories.

However, if non-uniform variation is present in instances of the real-time event being modeled by the trajectories, DTW is a more appropriate method. This is the case with speech recognition. For example, the word "one" can be pronounced as "wuhn", or as "wuuuuuuuuuuhn". Notice that time is not stretched uniformly in this case; the vowel sounds are stretched out, while the consonants remain unchanged. The UTW algorithm would interpret the second trajectory as "wwwwuuuhnnnn". DTW is better suited for compensating non-uniform time differences "on-the-fly".

One could argue that DTW should always work at least as well as UTW, if not better in dynamically stretched cases. Unfortunately, this may not always be the case because the DTW algorithm is dependent on several parameters (e.g. stretch factors) whose optimal values are highly data-dependent, whereas UTW depends only on the resolution of the segmentation chosen. A trade-off exists between error induced by suboptimal parameters and error induced by uniformly distributing non-uniform time distortions.

2.3 Summary

The FSTNN is a powerful pattern recognition tool which has been used successfully in SOI and Automatic Target Recognition. However, its use is limited to problems where independent test points are to be classified. When a sequence of test points must be classified as a group, the standard FSTNN algorithm can yield erroneous results. Neiberg and Casasent developed several methods for testing a set of sequential test points, but these methods merely consolidate the independent test point distances into a single metric. A point's position in the sequence should affect the way its individual distance measurement to a trajectory is calculated.

Two solutions to this problem are proposed. Dynamic Time Warping limits the portions of the training trajectory to which a test point can be matched based upon the where the previous points in the test sequence project. This algorithm allows "on-the-fly" time stretching of the trajectories with respect to each other. The other solution, Uniform Time Warping, segments both the training trajectory and test trajectory into segments of equal length and computes the

distance from corresponding segment endpoints. Both of these methods prevent a test point from matching to a portion of the training trajectory more previous in time than the previous test points.

III. Methodology

This chapter explains the data and FSTNN algorithms in further detail, emphasizing how the methods are implemented in the experiments of this thesis.

3.1 Simulated Data

The satellite image data was generated by Brandstrom using SatTools, a program written by Phillips Lab for AMOS, and are considered to be sufficiently realistic for the purposes of the SOI problem [4]. The images represent the TD81 satellite model in low earth orbit (orbital period of 90 minutes). The orbital plane is shifted by about 1° per day, resulting in the satellite passing over a given point at the same time each day; a sun-synchronous orbit [3].

The normal class of satellite orbits are nadir-pointing and 3-axis stable. The satellite should always be pointing towards the earth's center, and any roll, pitch, or yaw with respect to this position is considered anomalous. Figure 8 shows the first five images of three sequences. The first two represent normal orbits; a smooth pose transition can be followed from top to bottom. The anomalous sequence does not display a smooth, gradual pose transition, but rather appears to be tumbling.

3.1.1 Data Sets. The data is broken into two sets. The first set contains 100 20-image sequences. The images are taken at ten second intervals, and each sequence represents a different orbital pass with respect to Maui, the observation point (Figure 9). The set consists of 70 normal and 30 anomalous sequences.

Brandstrom used the first data set to test various choices of image processing techniques and features. He then took the most promising results and tested them on a second data set. The second data set was generated for the purpose of obtaining the best possible classification rates, using more precise orbital parameters, only descending passes, and image intervals of eight seconds (Figure 10). The second data set contains 46 normal and 23 anomalous sequences.

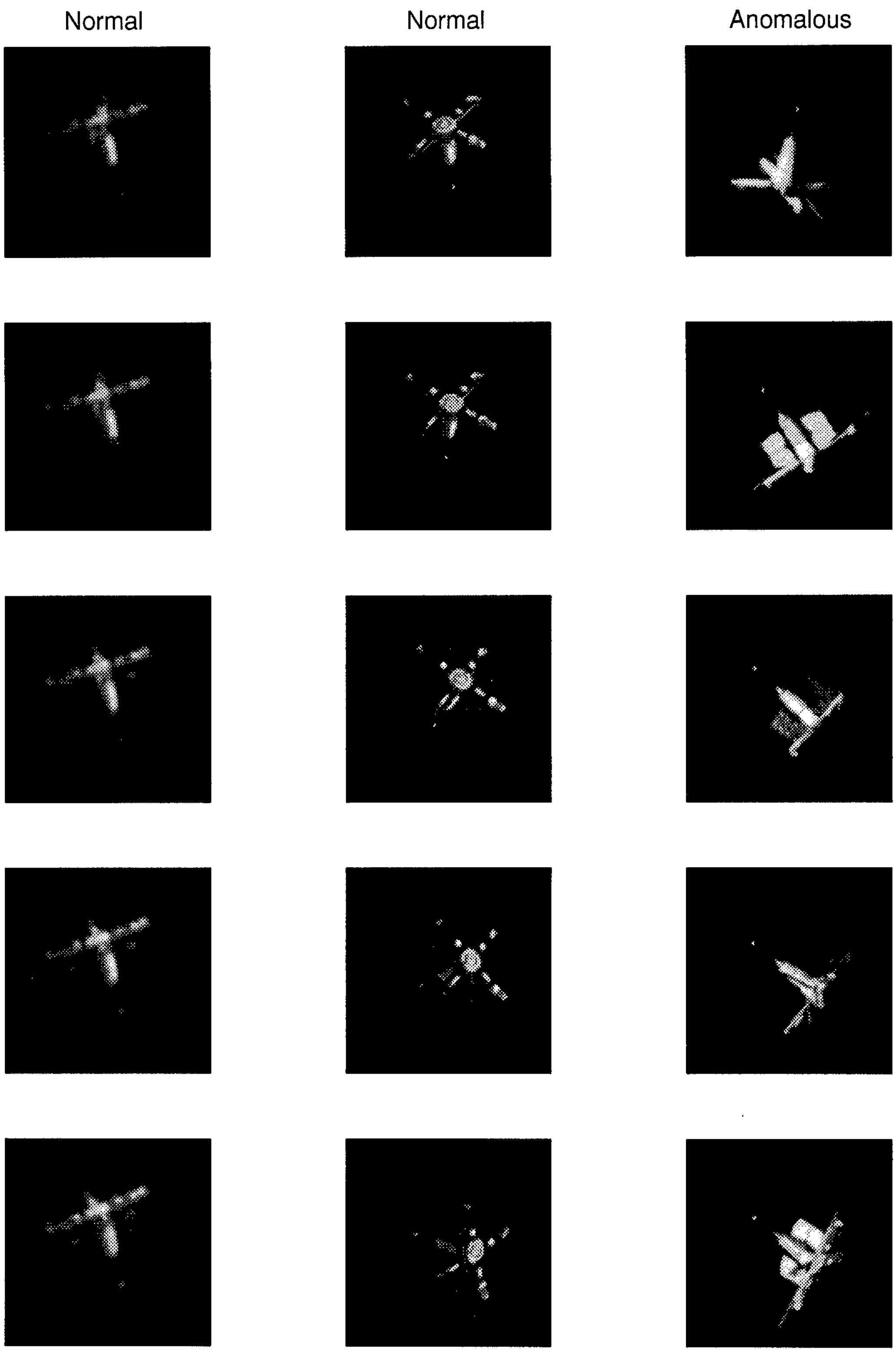


Figure 8. Samples of normal and anomalous image sequences. [4]

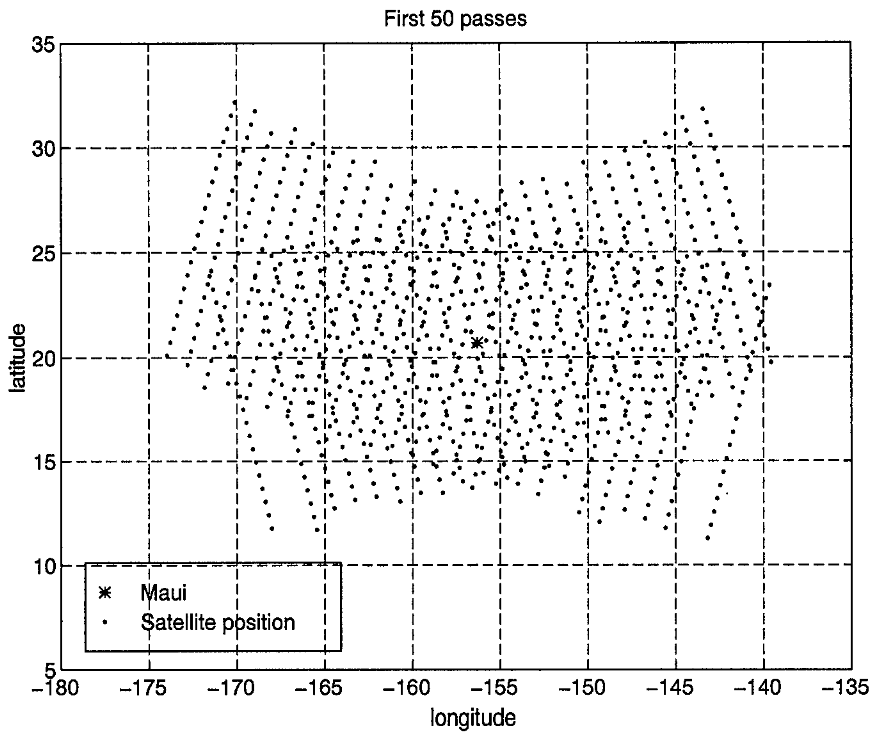


Figure 9. First set of training data. [4]

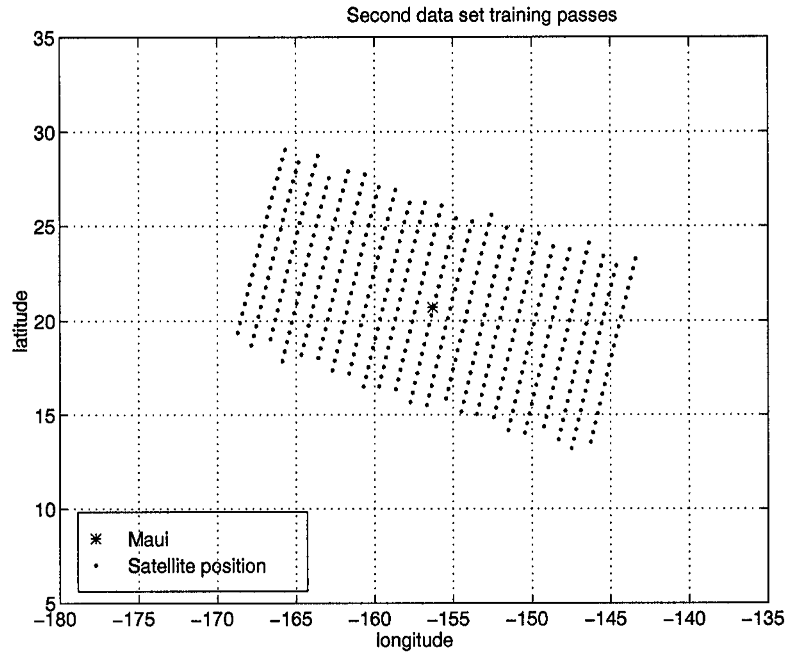


Figure 10. Second set of training data. [4]

3.1.2 Distortion. While testing the three algorithms on the pristine image data will yield a valid comparison of their relative effectiveness, these results do not constitute a valid solution to the SOI problem since actual image data obtained by AMOS has been degraded by noise, atmospheric turbulence, and the optics used. Additional tests must be run on degraded images in order to determine if (a) the effectiveness of the three methods degrades equally with distortion, if at all, and (b) the FSTNN is a realistic solution to the SOI problem. HYSIM3 is a simulation tool used to apply the distortion described in Sections 3.1.2.2 and 3.1.2.3 [17]. While the physics behind the calculation of these effects are somewhat beyond the scope of this thesis, a brief overview of the concepts are presented to convey a general understanding of how HYSIM3 is used to degrade the images.

3.1.2.1 Noise. Two types of noise are of primary concern when imaging satellites for the SOI problem: photon and receiver noise. Photon noise is caused by the random arrival times of the photons in the wave front, and is generally modeled as a Poisson arrival

process [18]. This type of noise usually becomes significant only in low-light conditions. Since this is not the case with the simulated images used in this thesis, this type of noise was not modeled.

(a) (b)

Figure 11. Comparison of (a) pristine image and (b) the same image with Gaussian noise.

Receiver noise is induced by the components of the imaging equipment and is Gaussian in nature. This type of noise was modeled adding a matrix of the absolute value of normally distributed numbers with a mean of 0 to the image matrix. The variance of these numbers was set at 5% of the highest value in the image matrix. Figure 11 shows an example of this level of noise. Each image in the data set was corrupted with an independent realization of receiver noise.

3.1.2.2 Atmospheric and Optical Distortion. The main cause of distortion when imaging an object in space is atmospheric turbulence. The Fried parameter, r_0, is used to quantify the amount of atmospheric turbulence present and represents the effective diameter of a diffraction limited lens that would yield an equivalent resolution with no turbulence present [22]. As one would expect, r_0 is not constant with respect to time, location, or even the various

strata of the atmosphere. For example, Figure 12 shows typical values of r_0 measured from Capella, United Kingdom on 26 December 1993 [1].

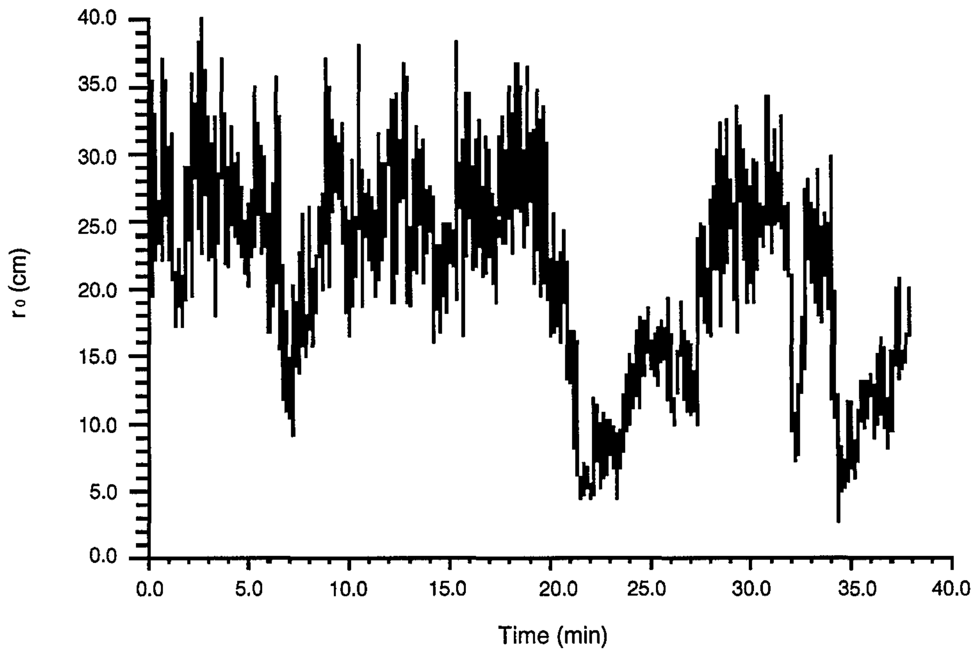


Figure 12. Fried parameter measured in Capella, United Kingdom. [1]

Optical distortion is induced by diffraction caused by the lens of the imaging telescope, which filters out a certain portion of the high-frequency components of the image. The degree of degradation is dependent upon the optics used and the imaging wavelength.

Both the atmospheric turbulence and the lens distortion are characterized in the Optical Transfer Function (OTF), which is an autocorrelation ($\star$) of the optical and atmospheric effects calculated in the Fourier space (f_x, f_y).

$$OTF(\vec{f}) = N_F^{-1}\left([P(\vec{f}\lambda d_i)exp(j\Phi(\vec{f}\lambda d_i))] \star [P(\vec{f}\lambda d_i)exp(j\Phi(\vec{f}\lambda d_i))]\right) \quad (1)$$

where $P(\vec{f}\lambda d_i)$ represents the pupil function, Φ is the wavefront phase in the pupil, and N_F is a constant which ensures $OTF(\vec{0}) = 1$ [18]. The pupil function, as discussed above, is a function of the component frequencies (f_x and f_y), the distance between the image and

pupil planes (d_i), and the imaging wavelength (λ). The pupil function has a value of 1 if the frequency combination is within the "pupil", as determined by λ and d_i, and a value of 0 if is not.

$$P(\vec{f}\lambda d_i) = \begin{cases} 1, & \text{if } (\vec{f}\lambda d_i) \in \text{Pupil} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The atmospheric turbulence is represented by the wavefront phase function, Φ, and varies randomly according to the level of r_0. HYSIM3 performs these calculations transparent to the user. The OTF is a matrix of weights which is multiplied with the 2-dimensional discrete Fourier transform (2-D DFT) of the pristine image. Figure 13 illustrates a typical OTF. Low frequency components in the center of the matrix remain largely unaffected while high frequencies are attenuated. Frequencies outside of the pupil domain are truncated completely.

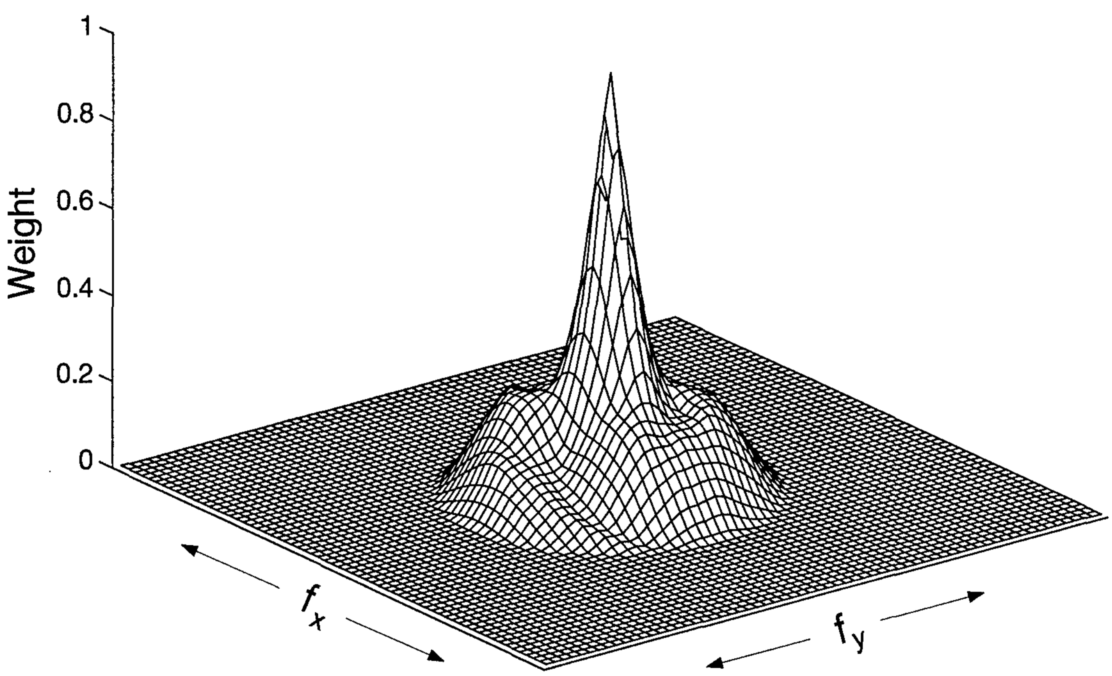


Figure 13. Optical Transfer Function

HYSIM3 was used to calculate OTFs based on the following parameters:

$$\lambda = 700\text{nm}$$

$$d_i = 1.6\text{m}$$

$$r_0 = .10, .25, .4\text{m for high, medium, and low turbulence, respectively.}$$

An OTF was generated for each of the three levels of turbulence and applied to each image in order to test the FSTNN algorithms at each level of degradation. Figure 14 shows examples of the three levels of degradation along with the pristine image. The first column shows the level of turbulence. The second column displays the OTF applied in the Fourier space, and the third column shows the 2-D DFT of the image after the application of the OTF. The last column shows the resulting optical view of the satellite under the given level of degradation. As turbulence increases, the OTF attenuates more and more of the high frequencies, resulting in a more blurred image. The average value of r_0 at AMOS in Maui is between 10 and 12 centimeters. Therefore, the "high turbulence" level used in these experiments can be considered typical of the imaging conditions at Maui. It should be noted, however, that AMOS uses an adaptive optics system to compensate for much of the atmospheric turbulence (see Section 3.1.2.3). Therefore, the high turbulence images produced for this thesis are in effect much worse than those actually obtained from AMOS.

3.1.2.3 Adaptive Optics. Even though an r_0 of 0.1 used in the high level of turbulence simulation is typical of the atmospheric conditions at Maui, the actual images obtained by AMOS are significantly better than this because of the adaptive optics system used. When the wavefront from an object passes through the atmosphere, it is tilted due to normal refraction effects, and is deformed due to the turbulence. The refraction effects account for approximately 87% of the power of the wavefront aberrations, and can be easily removed with a tilted mirror [18]. This is typically the first step in the adaptive optics process. Since the only effect of wavefront tilt is a shifting of the image, the tilted mirror simply serves to center

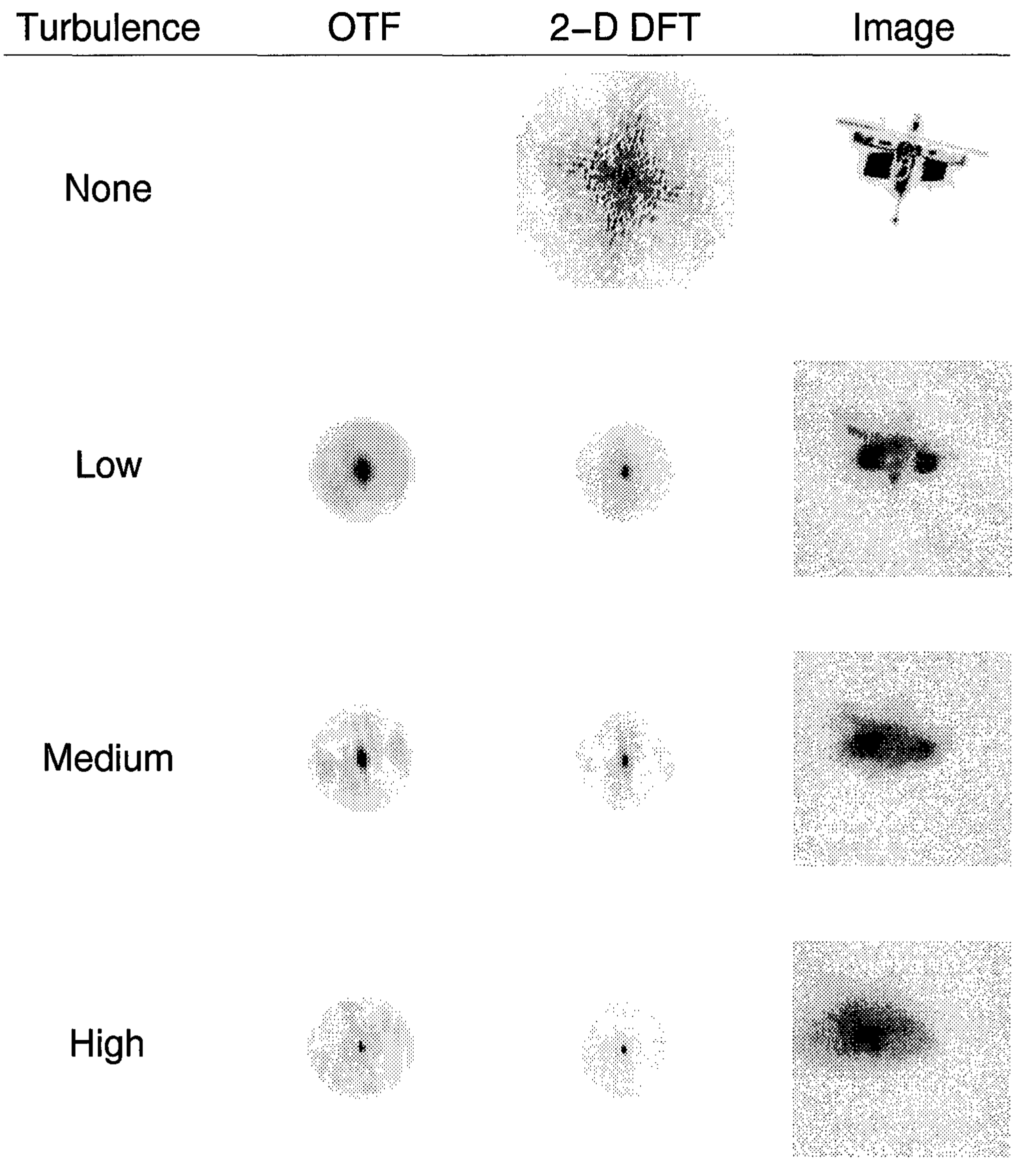


Figure 14. Satellite image under varying levels of turbulence.

the image on the imaging plane [22]. The remainder of the wavefront aberrations require a deformable mirror to correct, which is the main idea behind an adaptive optics system. Figure 15 shows a schematic of a typical adaptive optics system.

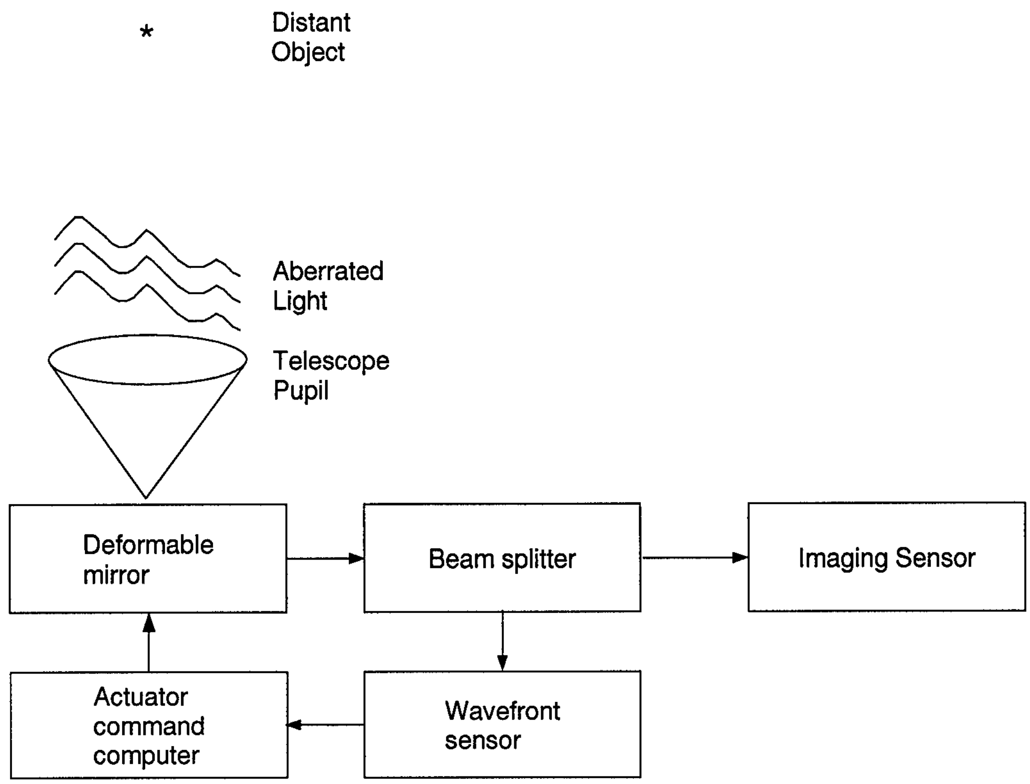


Figure 15. Schematic of Adaptive Optics System. [22]

The adaptive optics process involves a feedback loop which continuously detects the wavefront aberration and manipulates a deformable mirror to correct for the effects of turbulence. The deformable mirror has many actuators behind it which constantly adjust the surface of the mirror to the conjugate of the wavefront distortion, as calculated by the wavefront sensor. The accuracy of the correction is limited by the resolution (i.e. number and spacing) of the actuators. Thus, these systems cannot remove turbulence effects completely.

Modern adaptive optics systems, such as that of AMOS, often combine the deformable mirror concept with an image post-processing technique known as "speckle imaging". This process involves taking many snapshots of the image and using them to deconvolve the

turbulence effects from the object (Figure 16). Sometimes a point light source (i.e. a star) is imaged simultaneously to obtain a purer measurement of the atmospheric effects.

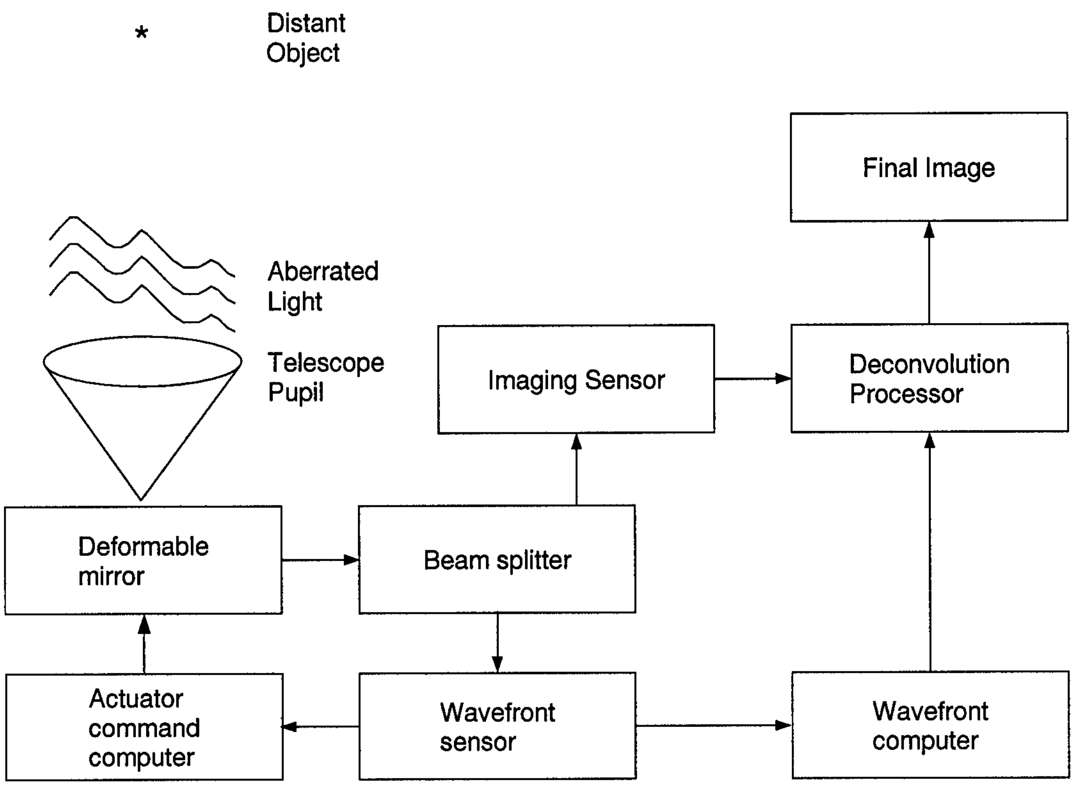


Figure 16. Schematic of Hybrid Adaptive Optics System. [22]

Because the satellite is constantly moving and the atmospheric effects are continuously changing, a limit on the number of snapshots that can be taken exists. For this thesis, HYSIM3 was used to simulate the effects of atmosphere as viewed through a hybrid adaptive optics system. Each pristine image is used to simulate 20 snapshots taken 1 millisecond apart. Changes in satellite position and atmosphere over a 20 millisecond time frame are assumed to be insignificant. These images were produced with an r_0 of 0.10 in order to simulate typical imagery obtained from AMOS.

3.2 Feature Selection

8-wedge Fourier features were selected for this analysis. These are obtained by taking a two-dimensional discrete Fourier transform (2-D DFT) of the image. Since the original images are 128x128 pixels, the 2-D DFT of these images will result in a 128x128 matrix of energy values. This matrix will be left-right symmetric, so it can be truncated to a 128x64 element matrix. The matrix is then divided into eight wedges (Figure 17) and the magnitudes

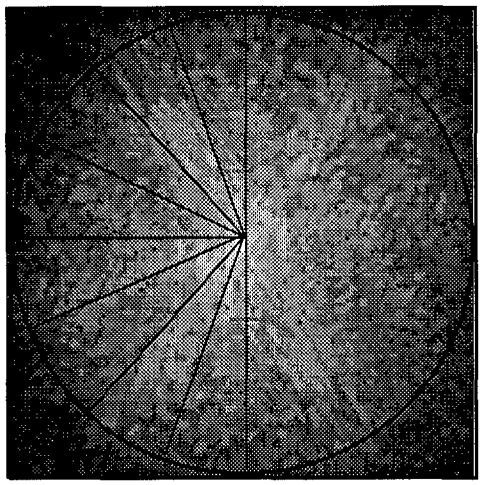


Figure 17. 8-wedge DFT Features. [4]

of the elements in each wedge are summed to produce the eight features. The features are then energy-normalized by dividing them by the number of pixels in their respective wedge. These features are good for this type of problem because they are shift, brightness, and scale invariant [6].

Figure 18 displays the process of degrading the images and obtaining the features. Images are obtained as 128x128 matrices. However, HYSIM3 required 512x512 image matrices, so the image is padded with 0's to reach the proper scale. Then the 2-D DFT of the image is taken and multiplied element-by-element by the OTF. The inverse of the DFT is then calculated to recover the image (now blurred) and the center 128x128 image is extracted.

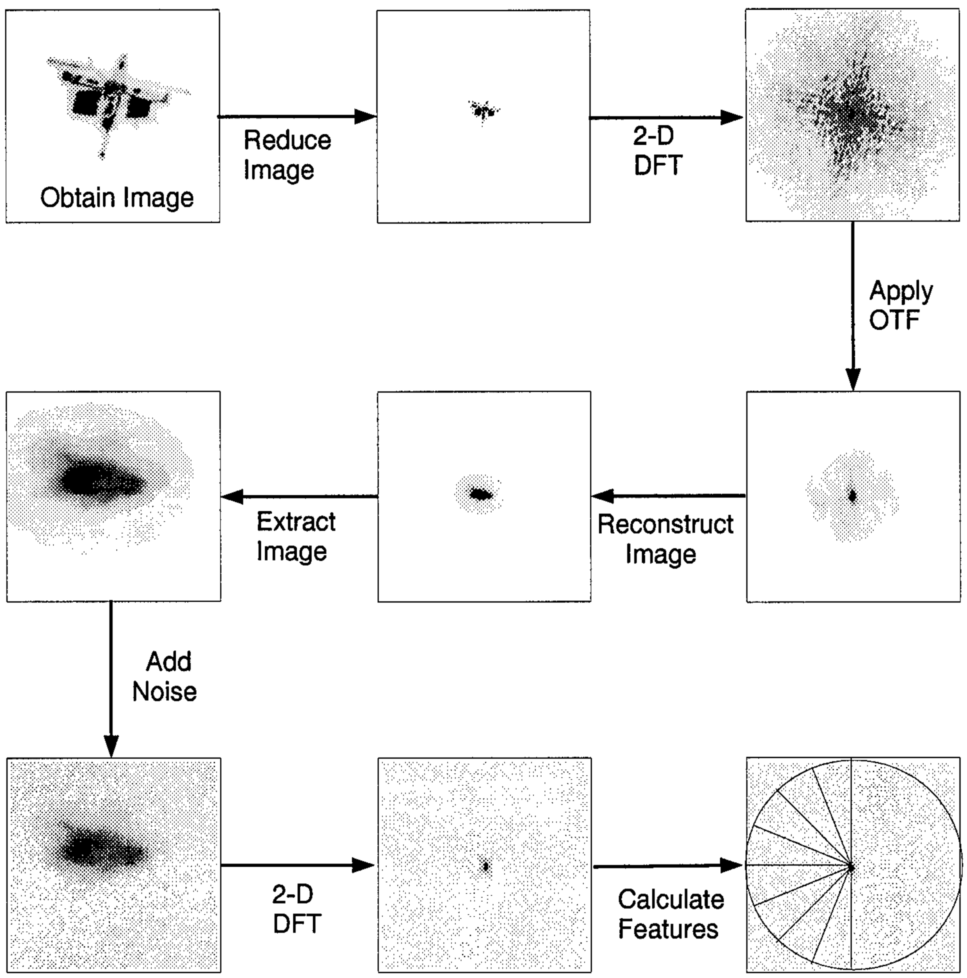


Figure 18. Process for degrading images and obtaining features.

After adding noise, the 2-D-DFT is performed again and the 8-wedge Fourier coefficients are calculated.

3.3 Feature Space Trajectory Neural Network

3.3.1 Distance Calculation. The basic FSTNN algorithm as originally developed by Neiberg and Casasent finds the Euclidean distance from a test point to each training, or known, trajectory. It accomplishes this by finding the distance of the test point to each link in a training trajectory. The smallest of these distances is the distance of the test point to that trajectory.

Let $T =$ Set of known vertices forming a training trajectory

$t_i =$ Training vertex

$P =$ Set of unknown vertices forming a test trajectory

$p_i =$ Test vertex

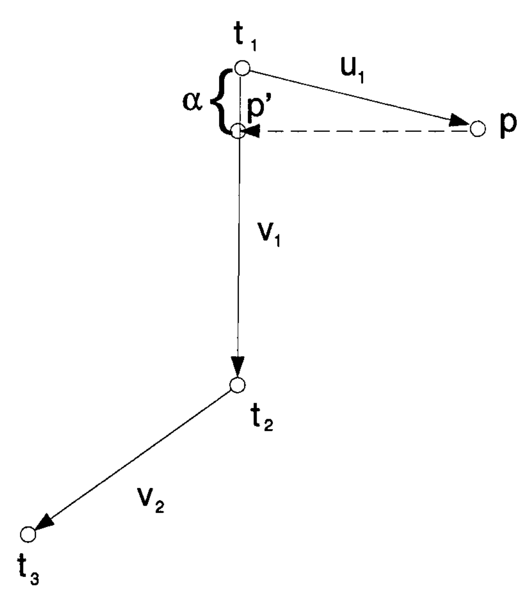


Figure 19. Projection of point p onto FST.

Given a test point p, let u_i be the unit vector with direction from t_i to p, and v_i be the vector from t_i to t_{i+1} (Figure 19).

$$u_i = t_i - p \tag{3}$$

$$v_i = (t_{i+1} - t_i)/l_i \tag{4}$$

where $l_i = ||t_{i+1} - t_i||$ is the length of vector v_i.

Now let p' be the projection of p onto v_i. Stated another way, let α be the projected length of u_i onto v_i.

$$\alpha = u_i \cdot v_i \tag{5}$$

The distance from p to p' is calculated in one of three ways depending upon the value of α. First, if α is less than zero, then p' does not lie on v_i and d_i is simply the distance from t_i to p .

$$d_i = ||t_i - p|| \tag{6}$$

If α is positive, but greater than l_i, then p' also does not lie on v_i and d_i is the distance from p to t_{i+1}. This is equivalent to the first case for the link v_{i+1}.

Finally, if α is positive but less than l_i, then p' lies on v_i, as shown in Figure 19. In this case, d_i is the distance from p to p'.

$$d_i = ||p - p'|| \tag{7}$$

$$d_i^2 = ||p - p'||^2 \tag{8}$$

$$d_i^2 = (p - p') \cdot (p - p') \tag{9}$$

$$d_i^2 = (p - \{(1 - \frac{\alpha}{l_i})t_i + \frac{\alpha}{l_i}t_{i+1}\}) \cdot (p - \{(1 - \frac{\alpha}{l_i})t_i + \frac{\alpha}{l_i}t_{i+1}\}) \tag{10}$$

Letting $a = 1 - \frac{\alpha}{l_i}$ and $b = \frac{\alpha}{l_i}$,

$$d_i^2 = (p - at_i - bt_{i+1}) \cdot (p - at_i - bt_{i+1}) \tag{11}$$

$$d_i^2 = p{\cdot}p - ap{\cdot}t_i - bp{\cdot}t_{i+1} - ap{\cdot}t_i + a^2t_i{\cdot}t_i + abt_i{\cdot}t_{i+1} - bp{\cdot}t_{i+1} + abt_i{\cdot}t_{i+1} + b^2t_{i+1}{\cdot}t_{i+1} \tag{12}$$

$$d_i^2 = p \cdot p - 2ap \cdot t_i - 2bp \cdot t_{i+1} + a^2t_i \cdot t_i + 2abt_i \cdot t_{i+1} + b^2t_{i+1} \cdot t_{i+1} \tag{13}$$

Letting $c_{i,i+1} = t_i \cdot t_{i+1}$,

$$d_i^2 = p \cdot p - 2ap \cdot t_i - 2bp \cdot t_{i+1} + a^2c_{i,i} + 2abc_{i,i+1} + b^2c_{i+1,i+1} \tag{14}$$

The distance, d_i, to each link in the training trajectory is calculated, and the minimum distance represents the distance of the test point to the trajectory. Neiberg and Casasent repeated this algorithm over several training trajectories, where the test point was classified according to the trajectory with the smallest distance to the closest link.

3.3.2 Standard FSTNN. For the SOI problem, Brandstrom repeated the previous algorithm for several test points, and summed the minimum d's for all of the test points to get a metric for the distance of the test set P to the training trajectory T.

$$D = \sum_{j=1}^{J} min\{d_{j,i}\} \tag{15}$$

where I is the number of links in the training trajectory T, and J is the number of test points in P.

However, when Brandstrom's methodology is repeated in this thesis, the mean distance, rather than the sum, will be used. This metric allows for test sets of different lengths to be compared. While all of the trajectories used in this thesis are in fact the same length, this metric does have more general applicability.

$$D = \frac{1}{J}\sum_{j=1}^{J} min\{d_{j,i}\} \tag{16}$$

3.3.3 FSTNN with Dynamic Time Warping. The implementation of DTW into the FSTNN is not very complicated. The distance calculation is the same as in the previous section, but we limit the number of training links to which the comparison is made and adjust the distances by the appropriate stretch factors. The adjustments made affect only the comparisons. Once the classification is made, the unadjusted distance is assigned to d_i. The stretch factors used are 2.0 for skipping a link and 1.5 for staying on the same link, as shown in Figure 5. Since the images were taken at constant time intervals, a large degree of unsynchronization is not anticipated. The need to skip a link is expected to occur less frequently than the need to stay at the same link. Still, the selection of these stretch factors was rather arbitrary and an analysis to find their optimal values for this problem is proposed for future research (Section 5.4.5).

3.3.4 FSTNN with Uniform Time Warping. In the case of uniform segmentation, the number of points of comparison are chosen as a multiple of the number of links in the trajectories (the "segmentation factor"). For example, if the test trajectory has 19 links and a segmentation factor of 3 is chosen, the algorithm will find $19 * 3 + 1 = 58$ equidistant points along the training and test trajectories; an average of 3 per link. These are referred to as *projection points*, and the distance metric is simply the average Euclidean distance between the corresponding projection points on the training and test trajectories. A segmentation factor of 3 was used in the experiments of this thesis. An average of 3 points per link is considered to be sufficient coverage of the trajectory. Large segmentation factors increase the computational time required to run the tests.

Let

$$\vec{X}^{<T>} = \text{Projection points on } T \tag{17}$$

$$\vec{X}^{<P>} = \text{Projection points on } P \tag{18}$$

$$N = \text{Number of projection points (same for } P \text{ and } T) \tag{19}$$

Then

$$d_n = \| d_n^{<T>} - d_n^{<P>} \| \tag{20}$$

$$D = \frac{1}{N} \sum_{n=1}^{N} d_n \tag{21}$$

3.4 Classification Threshold

Because anomalous trajectories are not a class with a discrete or explicitly definable set of members, classification of anomalous trajectories cannot be based upon the distance to other known anomalous trajectories, but rather by their distance to known normal trajectories.

To accomplish this, the distance of every normal trajectory to the closest normal trajectory is measured. Then the distance of every anomalous trajectory to the closest normal trajectory is computed. Hopefully, the two sets of distances are linearly separable so that a threshold distance can be chosen such that all of the normal trajectories are beneath the threshold and all of the anomalous trajectories are above the threshold. If some overlap exists, the threshold is chosen in one of two ways: to either maximize the probability of detecting anomalous sequences (P_D) or minimize the probability of misclassifying a normal sequence (i.e. probability of false alarm, P_{FA}). If the threshold is chosen as the smallest distance from the anomalous set, $P_D = 100\%$ and the FSTNN will successfully identify all anomalous sequences. However, any normal trajectories above the threshold will be considered anomalous also. On the other hand, if the threshold is chosen as the largest normal trajectory distance, $P_{FA} = 0\%$, but the FSTNN will classify any anomalous sequences below the threshold as normal.

For the SOI problem, the FSTNN will be acting as a screen, filtering out data which is obviously normal and passing along to the analysts only those image sequences which may be anomalous. Therefore, it is more desirable to have false alarms than to misclassify an anomalous sequence.

3.5 Feature Saliency

It would be useful to know which features contribute to the classification of the test trajectories and which do not. If some features do not contribute significantly, it may be possible to exclude them without decreasing the classification accuracy, allowing more rapid training and testing.

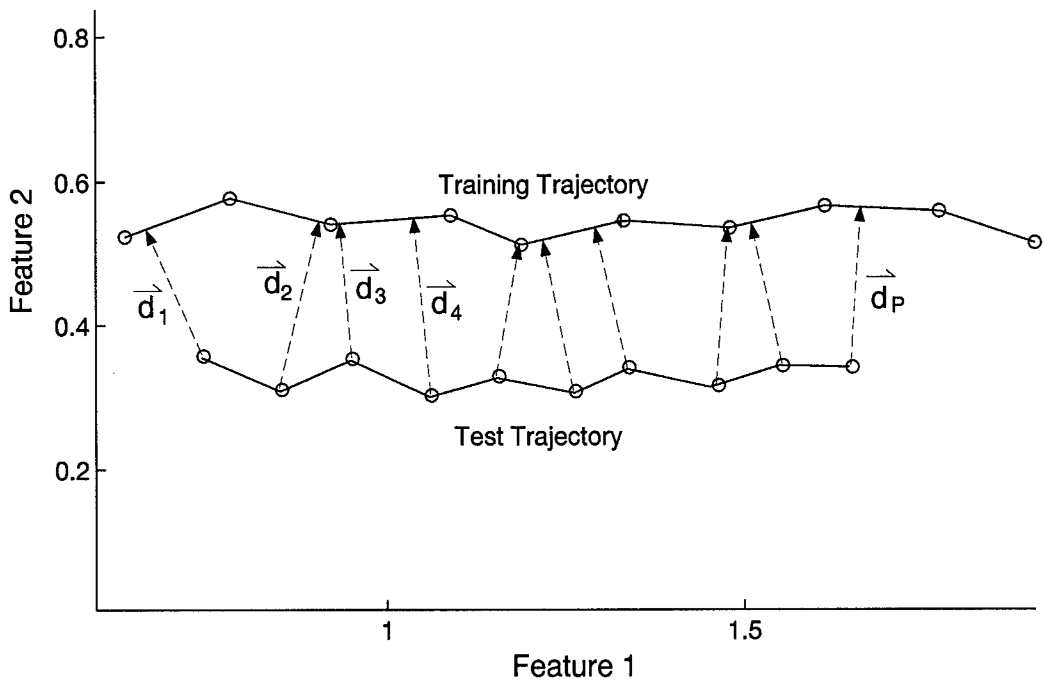


Figure 20. Example of distance vectors used in distance metric calculation.

Since the distance metric is an average Euclidean distance regardless of the method used, the contribution of each feature is simply the partial derivative of the distance with respect to that feature. Figure 20 shows an example of an FSTNN trajectory comparison. The $\vec{d_i}$'s are the distance vectors of each test point to the training trajectory. The distance metric becomes

$$D = \frac{1}{P}\sum_{i=1}^{P} ||\vec{d_i}|| \tag{22}$$

where P is the number of test points and therefore the number of $\vec{d_i}$'s. The contribution of a feature to a single $\vec{d_i}$ is the partial derivative of that $\vec{d_i}$ with respect to its component in the

direction of that feature.

$$||\vec{d_i}|| = \sqrt{d_{i,1}^2 + d_{i,2}^2 + d_{i,3}^2 + \cdots + d_{i,F}^2} \tag{23}$$

$$\frac{\partial ||\vec{d_i}||}{\partial d_{i,f}} = \frac{2d_{i,f}}{2\sqrt{d_{i,1}^2 + d_{i,2}^2 + d_{i,3}^2 + \cdots + d_{i,F}^2}} = \frac{d_{i,f}}{||\vec{d_i}||} \tag{24}$$

where F represents the number of features and f represents a particular feature. We can then define the derivative-based saliency metric as the sum of the absolute values of these partial derivatives with respect to any given feature.

$$R_f = \sum_{i=1}^{P} \frac{|d_{i,f}|}{||\vec{d_i}||} \tag{25}$$

The notation R is used for this metric since it is essentially the same as the Ruck Saliency Metric for an MLP computed at the decision boundary [19, 7]. Ruck proposed a saliency metric for the MLP by finding the gradient of the network's output with respect to its input features at various "pseudo-samples" throughout the feature space. Lee and Landgrebe proposed that all of the saliency information could be found by analyzing the gradient of the output at the decision boundary [8]. In fact, normal vectors to the decision boundary in a two-class problem are often approximated by finding, for each member of a class, the direction of the nearest member of the opposite class [20], which is exactly what the $\vec{d_i}$'s represent in the FSTNN.

A few modifications to this metric are necessary to apply it to the FSTNN. Using Equation 25 results in normalized vectors. However, since the $\vec{d_i}$'s are not all the same length, we would like to weight each $\frac{\partial ||\vec{d_i}||}{\partial d_{i,f}}$ by the length of the vector, $||\vec{d_i}||$. This leads to the convenient result:

$$R_f = \sum_{i=1}^{P} |d_{i,f}| \tag{26}$$

or

$$\vec{R} = \sum_{i=1}^{P} \left|\vec{d_i}\right| \tag{27}$$

where $\vec{R}$ is a vector of the new weighted Ruck saliency metrics for the features of the FSTNN. Thus, the saliency of the features is obtained by simply summing the $\vec{d_i}$'s. In problems like SOI, where multiple test sets are tested against multiple training trajectories, a single $\vec{R}$ can be computed by either averaging or summing the individual $\vec{R}$'s for the winner of each test.

Unfortunately, since the function of the FSTNN is to discriminate between normal and anomalous trajectories, the raw distance measure of a trajectory comparison does not contain the information necessary to make this distinction, but rather the distance of the anomalous trajectories *relative to the distance of the normal trajectories*. Therefore, the true saliency can be found by comparing the saliency metrics (as in Equation 27) from the anomalous class to the normal class.

$$\vec{S} = \vec{R}_A - \vec{R}_N \tag{28}$$

The features with the highest S values are those components of the distance metric which contribute most to the difference between normal and anomalous sequences. If any features do not contribute significantly, the FSTNN might be able to obtain the same classification results without them. Determining what is "significant" is obviously a subjective decision.

3.6 Summary

The data used for this thesis consist of two sets of simulated satellite image sequences generated by Brandstrom for use in his thesis. The first set contains 100 20-image sequences representing both ascending and descending passes, with the images taken 10 seconds apart. The second set contains 69 20-image sequences representing descending passes only, with the images taken 8 seconds apart. This set was designed to obtain better classification accuracy. In addition to the pristine images, four additional tests were run on degraded image sequences simulated using the HYSIM3 simulation tool. Three of these tests involved corrupting the images with one of three optical transfer functions to simulate atmospheric turbulence and optical pupil degradation. The final test was run in high-turbulence conditions with a hybrid adaptive optics (Hybrid AO) system similar to that of AMOS.

The features extracted from each image are 8-wedge energy normalized Fourier features. These features are shift, brightness, and scale invariant, but are sensitive to rotation. Since anomalous orbits are characterized by the presence of excessive rotations, these features should contain adequate discriminating information to distinguish normal and anomalous classes.

The data will be tested using three forms of the FSTNN: the standard version developed by Neiberg and Casasent, the DTW version, and the UTW version. The results of these methods will be compared by setting the classification threshold such that a P_D of 100% is achieved, and comparing the resulting P_{FA}.

Finally, an experiment will be performed to test the saliency of the features used. This will determine if any of the features do not contribute to the ability of the FSTNN to detect anomalous sequences.

IV. Results

The results of the various experiments are contained in this chapter. Most of these results will be in the form of receiver operating characteristic (ROC) charts, which show the relationship between P_D and P_{FA} for each experiment.

4.1 Classification Results

Figures 21-25 show the results for the tests on the pristine images, low-turbulence data, medium-turbulence data, high-turbulence data, and the simulated adaptive optics data. The first column represents data set 1 and the second column represents data set 2. The three rows represent the three methods used: the standard FSTNN, DTW, and UTW.

Since it is desirable to have a P_D of 100%, the main point of concern from these ROC curves is the maximum P_{FA}, which occurs when the P_D reaches 100%. Figure 26 displays a comparison of the maximum P_{FA} for each method over the range of degradation tested for data set 1. Figure 27 shows the same information for data set 2. The single points represent the maximum P_{FA} for the adaptive optics test from Figure 25.

Figures 26 and 27 show very clearly that DTW performs consistently better than the standard FSTNN algorithm. However, the UTW method seems to be worse in all cases except the high-turbulence case in data set 2, where it performs as well as DTW. This reveals a very unexpected result; in many cases, the classification accuracy improves as the turbulence increases. One would be tempted to infer from this that as the images get blurrier they become easier to recognize. However, it is important to remember that the FSTNN is not recognizing images, but rather the behavior of those images in time. For this, a large degree of detail may not be necessary.

Recall from Figure 9 that each normal trajectory represents a different orbital pass. This results in a different aspect view of the satellite for each normal pass. Therefore, the entire set of normal trajectories are all somewhat different from each other. However, as the images are blurred, the normal satellite sequences begin to look more and more like each other until they

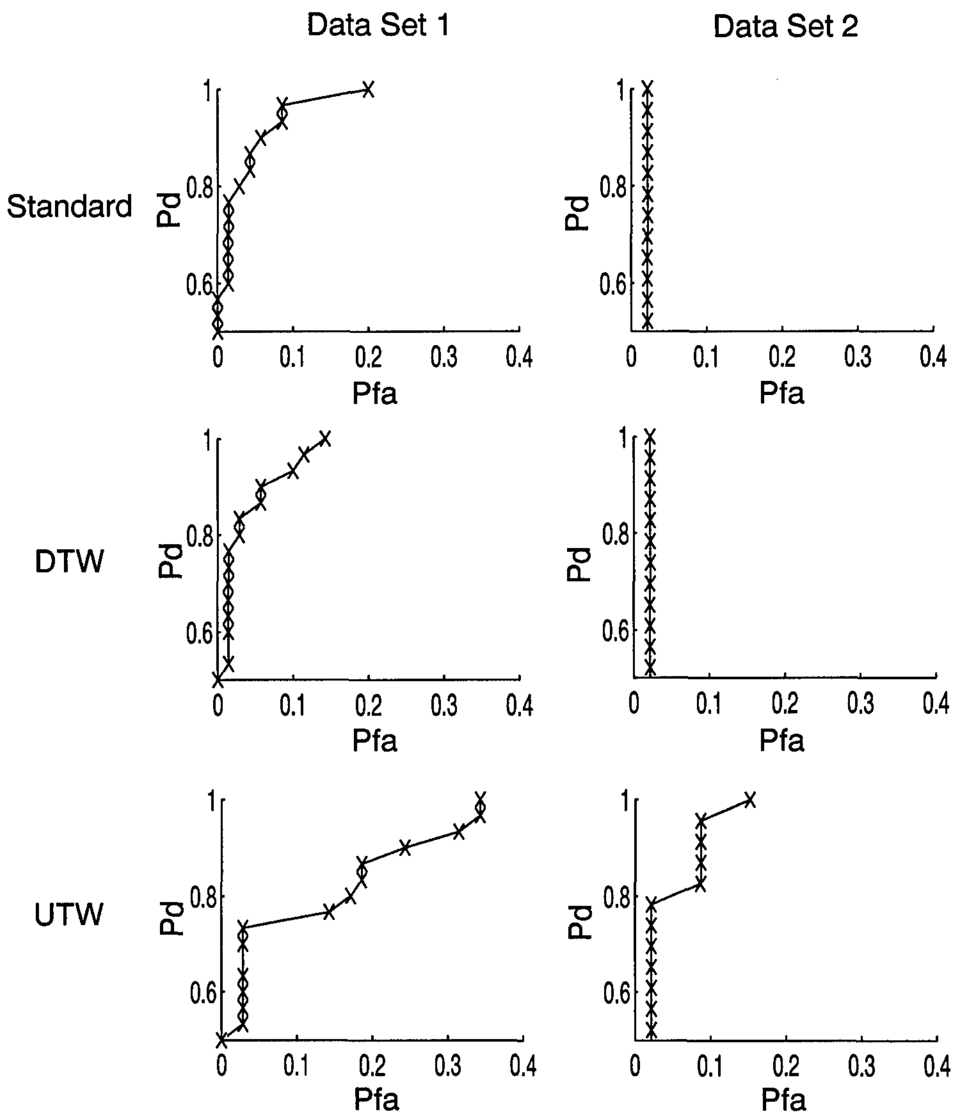


Figure 21. ROC curves for pristine satellite data.

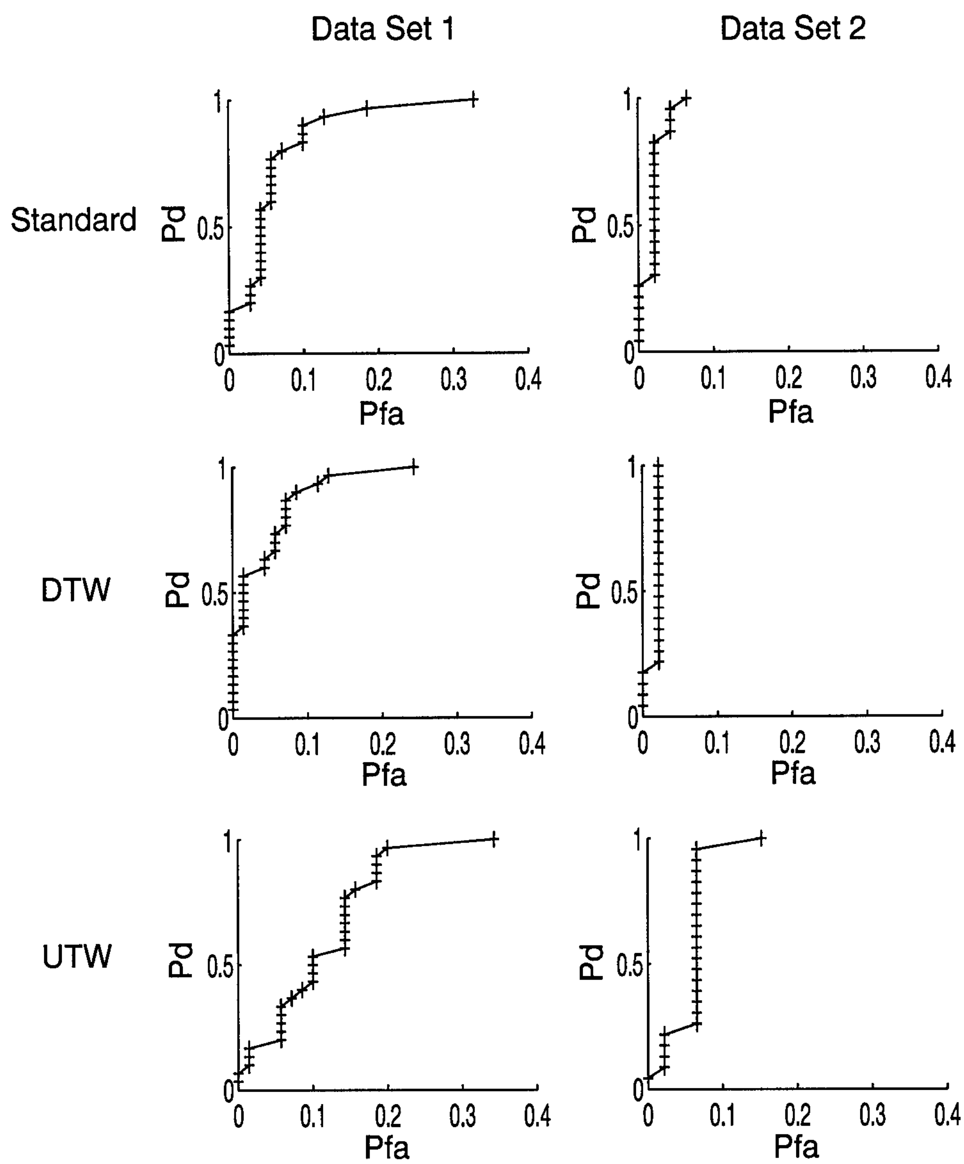


Figure 22. ROC curves for low-turbulence satellite data.

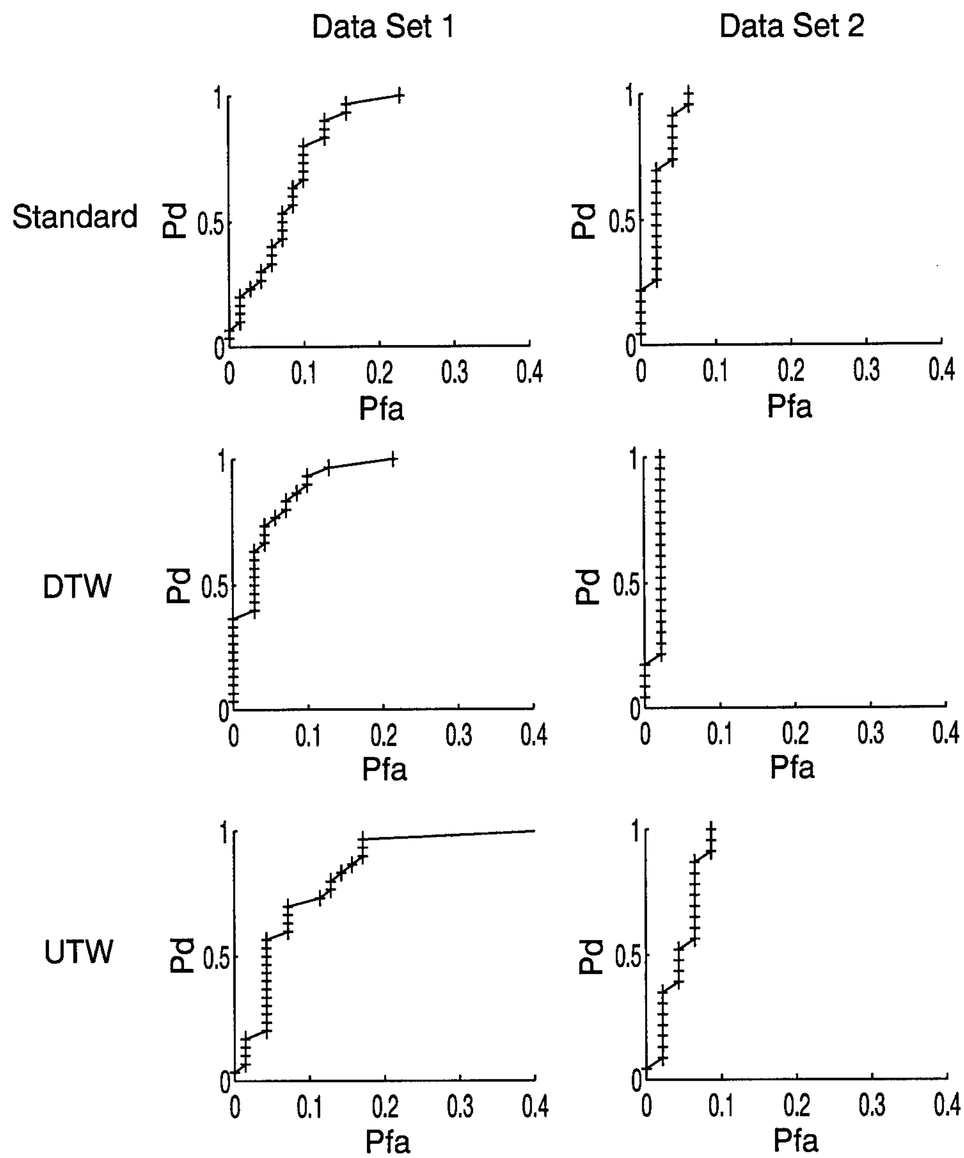


Figure 23. ROC curves for medium-turbulence satellite data.

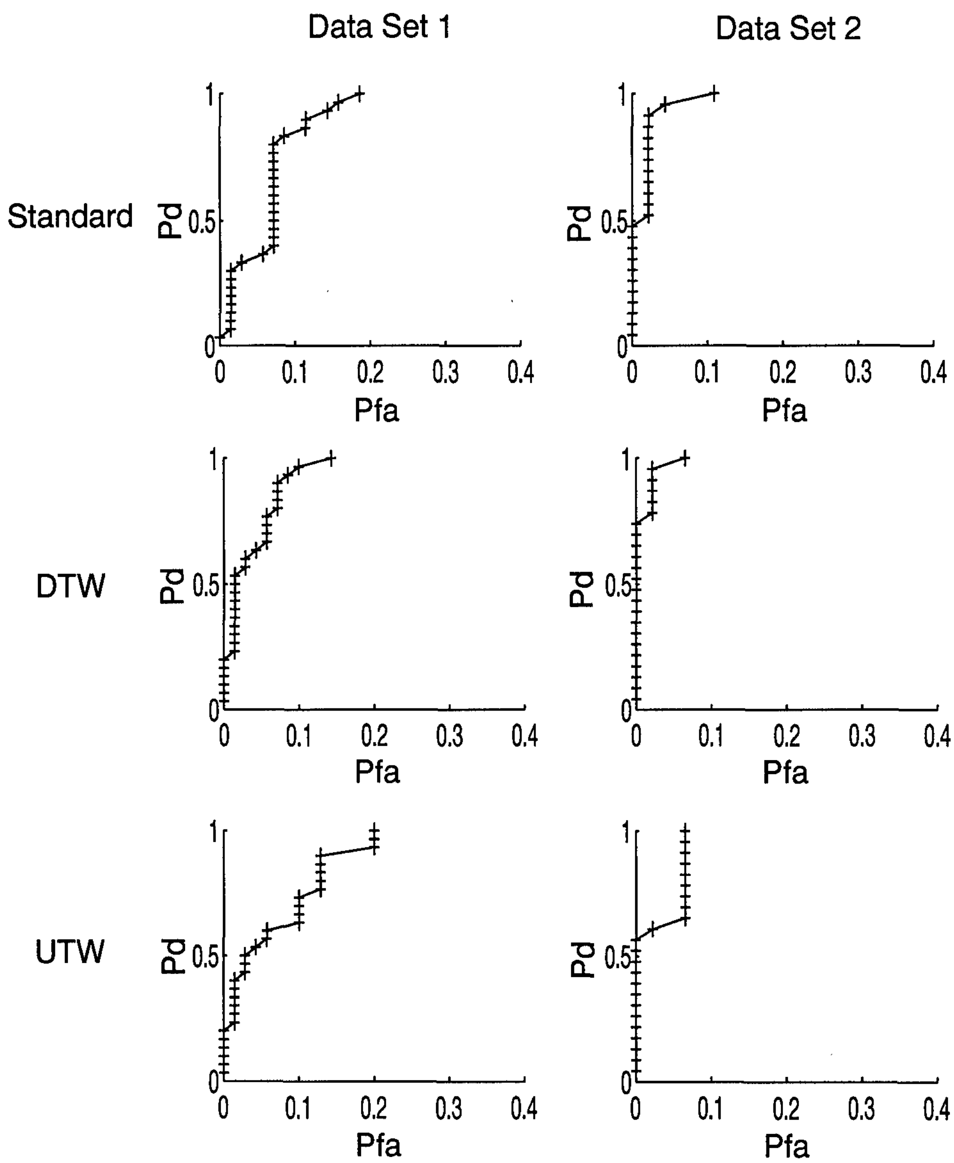


Figure 24. ROC curves for high-turbulence satellite data.

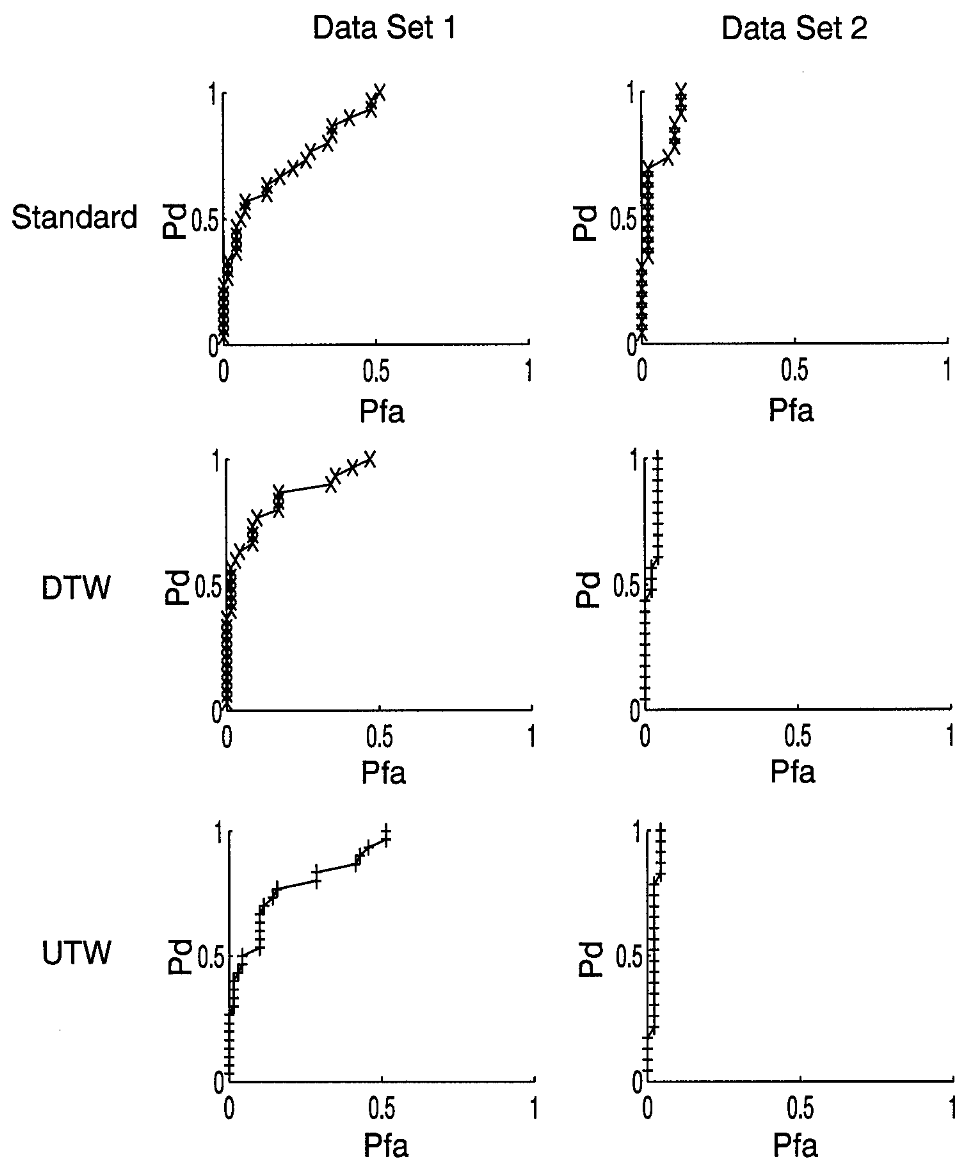


Figure 25. ROC curves for satellite data using hybrid adaptive optics.

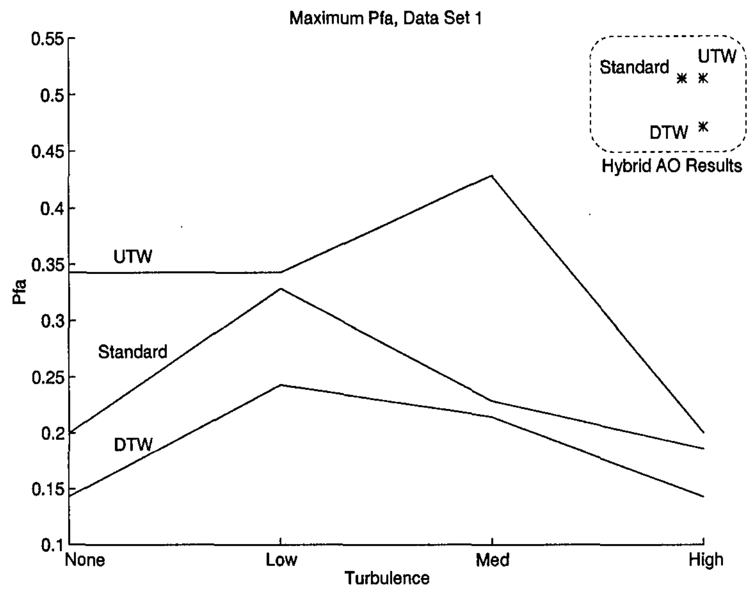


Figure 26. Maximum P_{FA} for data set 1.

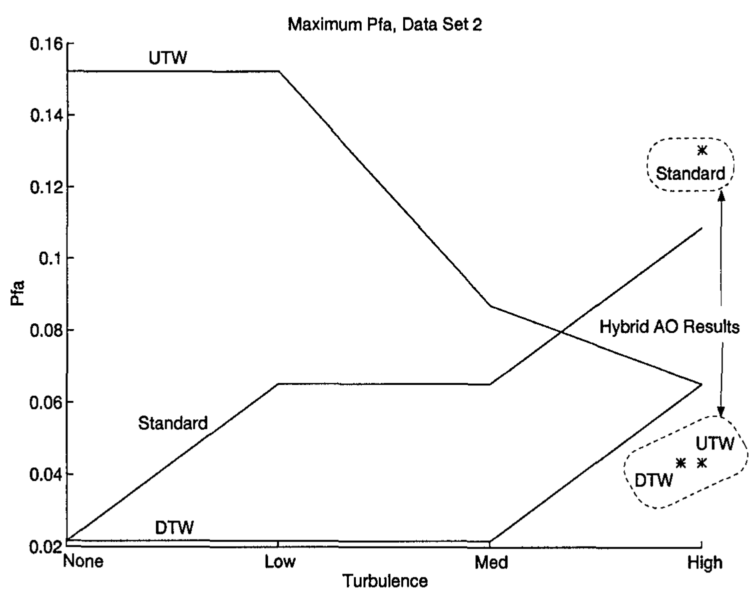


Figure 27. Maximum P_{FA} for data set 2.

are all simply elliptical blurs. Since the anomalous class is defined by abnormal rotations, it may still be possible for the FSTNN to detect these rotations in the blurred ellipses.

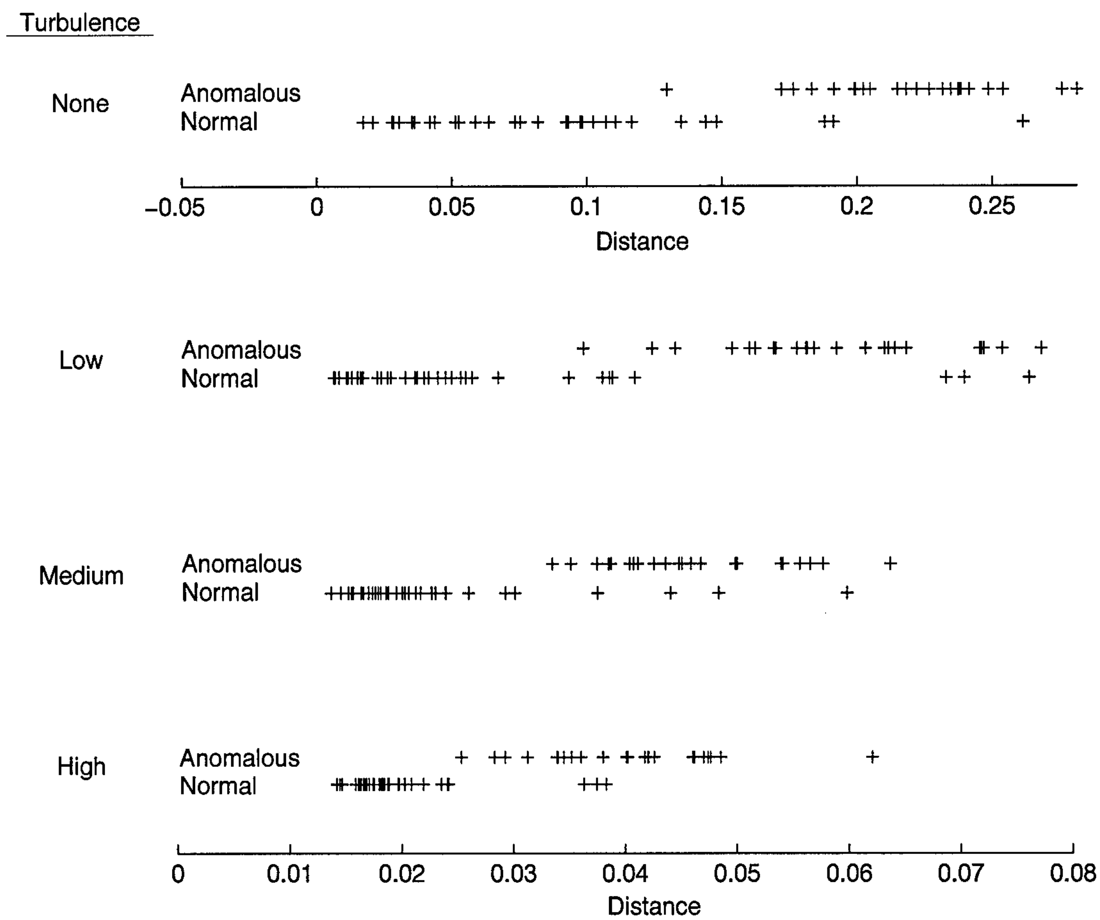


Figure 28. Example of distance distributions for varying levels of turbulence.

To demonstrate these effects, an example of the distributions of the normal and anomalous test distances are plotted in Figure 28. This chart represents the UTW method on data set 2, which exhibited the classification increase under turbulence. Notice that as the turbulence increases, the mean and variance of the distributions decreases, especially among the normal classes. The distribution of the pristine images is on a larger scale than those with turbulence added. As soon as the low level of turbulence is added, the normal and anomalous distributions decrease in scale by about 75%. They continue to shrink with more turbulence, but not as drastically. However, the FSTNN still maintains adequate separation of the normal and

anomalous classes. In this case, the reduction in the mean and variance of the normal class was greater than the reduction in mean and variance of the anomalous class, resulting in an increased classification accuracy.

The phenomenon of an increased classification accuracy with increased turbulence did not occur for every test. For example, the performance of the standard FSTNN and DTW deteriorated slightly for data set 2 as turbulence was added. Since these tests were run using all of the data and only a single realization of the OTF at each turbulence level, more tests are necessary to perform a statistical analysis of the results. Running these tests on the entire data set is very time consuming, so a random subset of the data was tested 20 times at each turbulence level. The reduced data set consisted of 20 normal and 20 anomalous sequences from data set 1, and each test utilized a different realization of the OTF at each level.

Figure 29 shows the results of these tests. The first three charts show the average maximum P_{FA} at each turbulence level for each of the three methods. The dashed lines denote two standard deviations from the mean. Chart (d) shows the mean maximum P_{FA} for each method. From these results it is clear that no statistical difference exists between the classification rates at the different turbulence levels for any method. The P_{FA} values are high because of the limited data used in these tests. Based on these 20 runs, there is also no statistical difference between the average classification results of the three methods. However, this does not imply that the methods are not different. Figures 30 and 31 show the maximum P_{FA} charts for each of the 20 runs. From these charts it is clear that for any particular set of data, the DTW and UTW algorithms will outperform the standard FSTNN algorithm, with only a few exceptions.

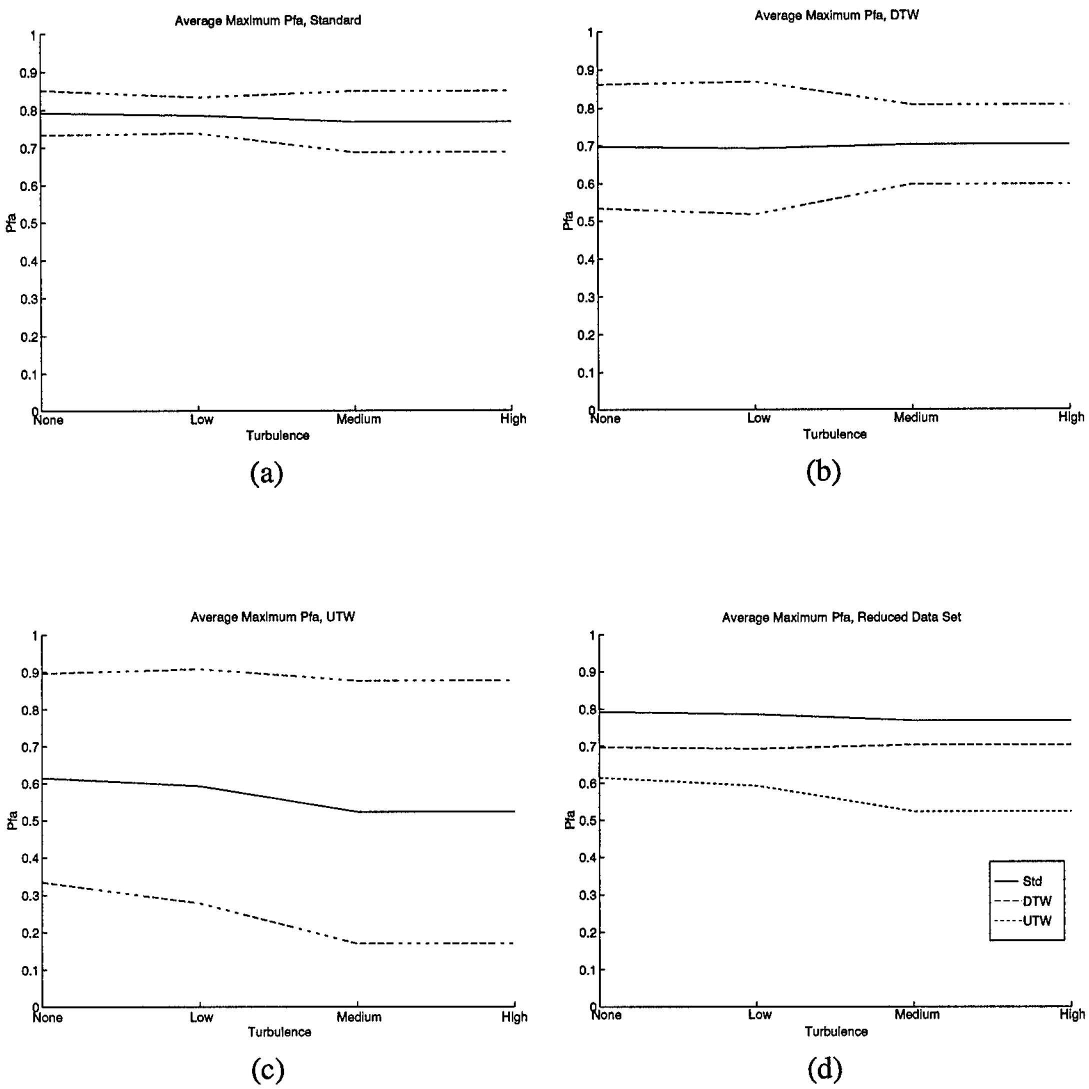


Figure 29. Results for 20 runs on reduced data set for (a) standard FSTNN, (b) FSTNN with DTW, and (c) FSTNN with UTW. Chart (d) shows average maximum P_{FA} for all three methods. Dashed lines in (a), (b), and (c) denote two standard deviations.

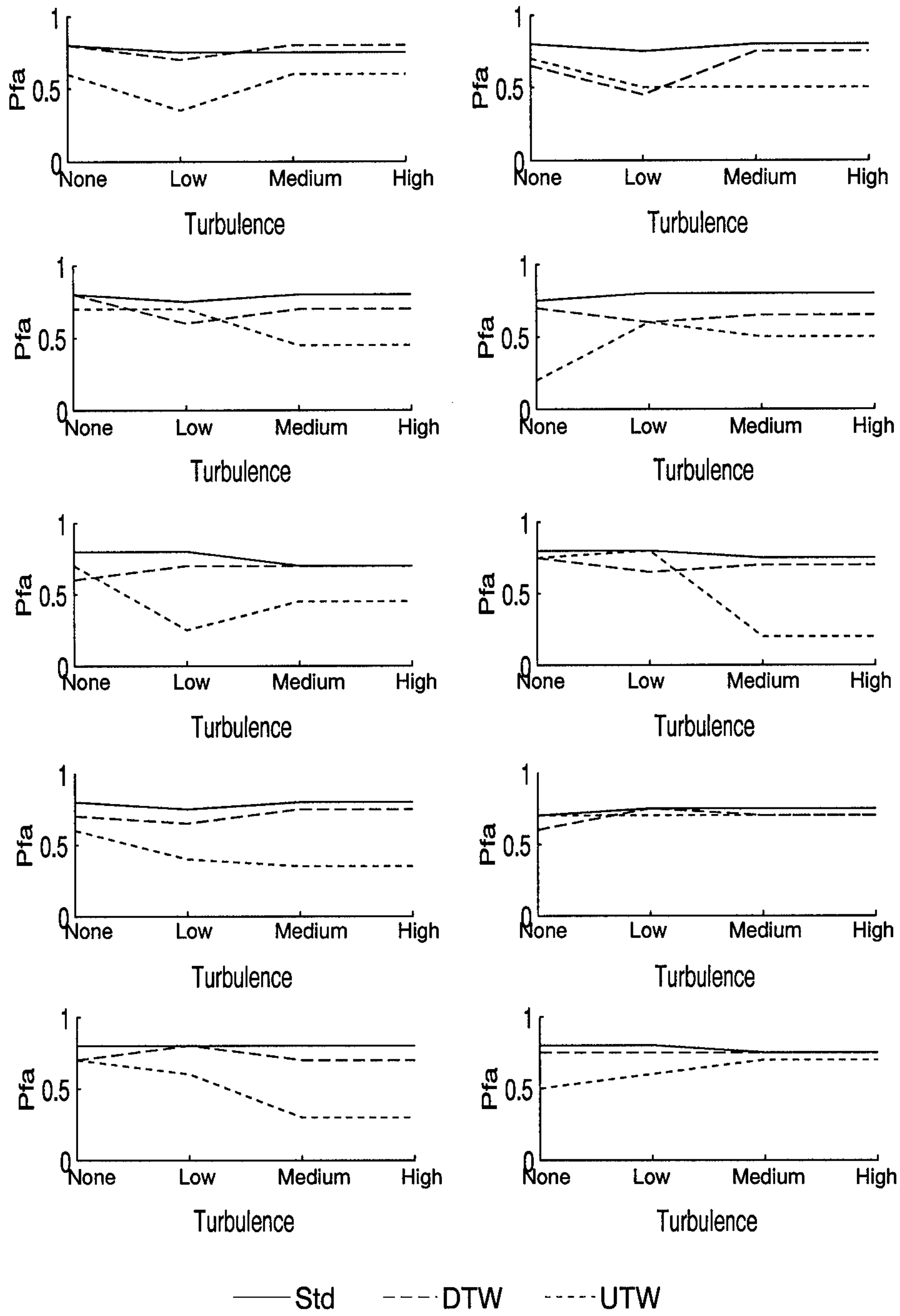


Figure 30. Maximum P_{FA} for first 10 runs.

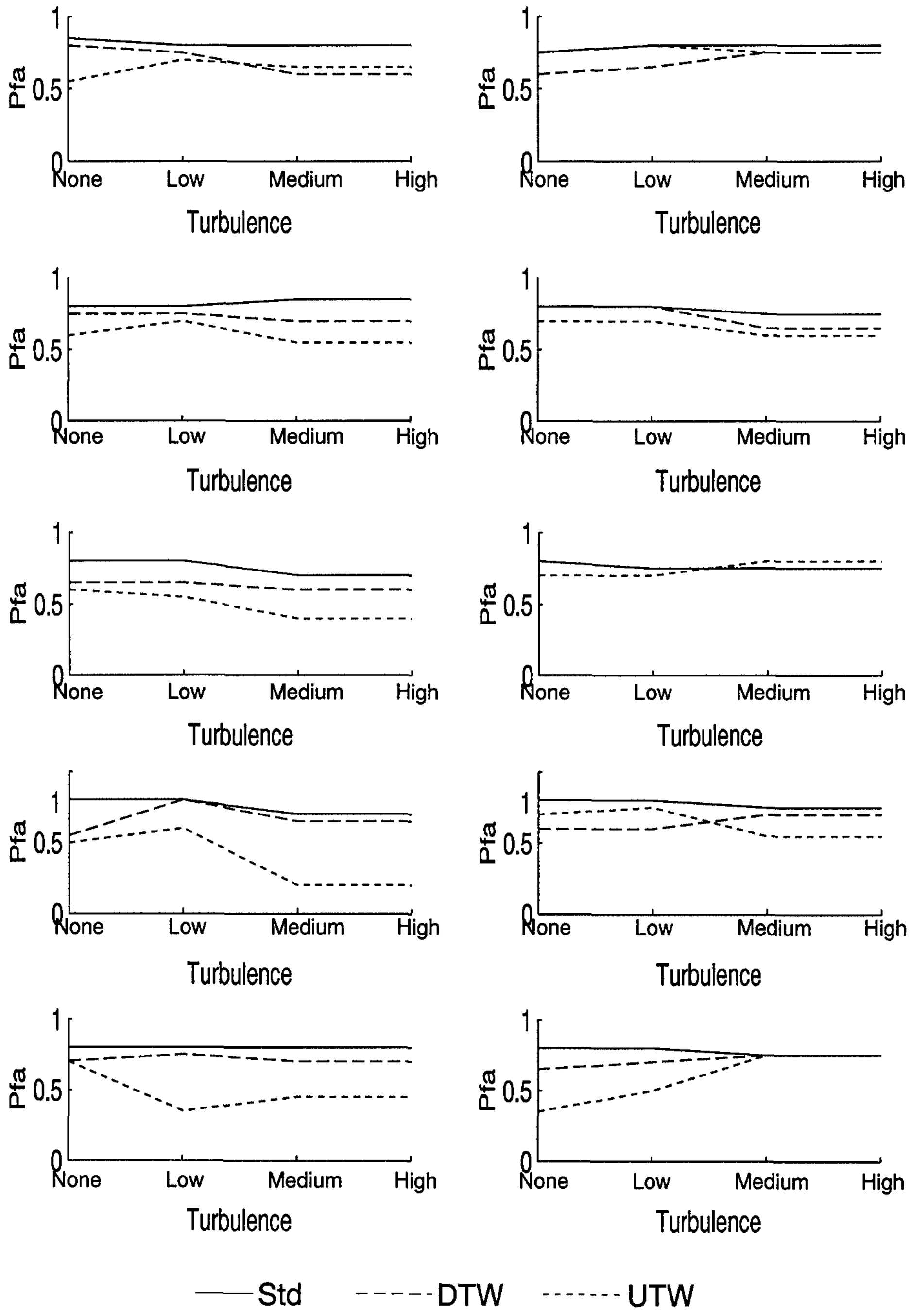


Figure 31. Maximum P_{FA} for last 10 runs.

4.2 *Feature Saliency*

A set of 30 normal and 30 anomalous trajectories were used to test the saliency of the features with each of the three FSTNN algorithms. Each sequence yields a single $\vec{R}$, which are averaged to form an overall $\vec{R}$ for each class, $\vec{R}_A$ and $\vec{R}_N$. The results are shown in Tables 1-3. The *% Total* column shows the S_f values as a percentage of the total sum. This allows for a better comparison between methods, since the S_f ranges are different for each.

Feature	S_f	*% Total*
5	0.0957	24.97
8	0.0933	24.34
2	0.0797	20.79
3	0.0605	15.78
7	0.0182	4.74
6	0.0179	4.67
4	0.0120	3.14
1	0.0060	1.58

Table 1. Feature saliency for standard method.

Feature	S_f	*% Total*
1	0.1890	25.09
8	0.1832	24.31
2	0.1802	23.92
5	0.0748	9.92
3	0.0497	6.60
4	0.0398	5.29
7	0.0280	3.72
6	0.0086	1.15

Table 2. Feature saliency for DTW method.

It is obvious from these tables that some of the features are not significant in distinguishing between normal and anomalous sequences. Furthermore, the choice of insignificant features is different for each method. The relative order of the features gives some insight into the time stretching being performed. For example, Feature 1 is not considered salient in the standard FSTNN algorithm. However, it is the most salient feature for the DTW algorithm.

Feature	S_f	*% Total*
3	0.6423	31.36
8	0.4496	21.95
5	0.3353	16.37
2	0.2268	11.07
6	0.1689	8.25
4	0.1172	5.72
1	0.0861	4.21
7	0.0219	1.07

Table 3. Feature saliency for UTW method.

This indicates that most if the "time stretching" being performed by the DTW algorithm is occurring in Feature 1.

A second test was run on this set of data with features 1 and 4 removed. These features are not very salient in the standard and UTW algorithms. The maximum P_{FA} for each method for both tests are shown in Table 4.

Method	*8 Features*	*6 Features*
Standard	0.6333	0.5000
DTW	0.5000	0.5000
UTW	0.6333	0.6667

Table 4. Comparison of maximum P_{FA} with 8 features and 6 features.

The ommision of Features 1 and 4 resulted in the UTW method misclassifying 1 additional normal trajectory. However, it had no effect on the accuracy of the DTW method, and the standard method actually improved. If the features removed were adding nothing more than noise, it is not surprising that the classification accuracy could improve. What is perhaps more perplexing is the lack of effect this had on the DTW method. Since feature 1 was deemed most salient, one would expect classification accuracy to decrease without it. The reason for this insensitivity may lie in the algorithm's dynamic nature. Feature 1 was most salient because it was in that dimension that time was stretched the most. Without that feature, the algorithm will perform its time stretches differently. For this reason, the saliency metric derived for the FSTNN may not be very reliable for the DTW algorithm.

4.3 Summary

Tests were run on data sets 1 and 2 at four levels of atmospheric turbulence: none, low, medium, and high. In addition, a test using simulated hybrid adaptive optics data was performed. Gaussian noise was also added to further degrade the images. Furthermore, a reduced data set of 20 normal and 20 anomalous sequences was tested for 20 different realizations of atmospheric turbulence at each level in order to find confidence bounds on the results. There was found to be no statistical difference in classification accuracy between the various levels of turbulence. In addition, DTW consistently outperformed the Standard FSTNN algorithm. UTW performs better than the standard algorithm on the average, but not consistently.

The metric for feature saliency was tested on a reduced data set and found to be an accurate gauge of saliency for the standard FSTNN and UTW, but questionable for the FSTNN with DTW because of its dynamic nature.

V. Conclusions

This chapter will summarize the main conclusions drawn from this research and will present several topics for future research in this area.

5.1 FSTNN Performance

Several sources of error exist in the FSTNN. Among these are feature selection, the linear trajectory approximations, and of course, improper synchronization of the test and training trajectories. This thesis has focused on improving the FSTNN's classification accuracy by reducing the error induced by improper synchronization. Two methods for reducing this error have been proposed and tested: a Dynamic Time Warping algorithm and a Uniform Time Warping algorithm.

It was known from Brandstrom's work that the FSTNN is a feasible solution to the SOI problem using images corrupted with shot noise. In addition to comparing the three types of FSTNNs on the pristine images, tests were also performed on images corrupted with various levels of atmospheric turbulence. A test was also run using the HYSIM3 simulation to emulate the hybrid adaptive optics system used by AMOS in order to test the "true" expected classification accuracy for these methods on actual AMOS data.

5.1.1 Standard FSTNN. The standard FSTNN, defined by Neiberg and Casasent's Vertex Sequence Method and used by Brandstrom on the SOI problem, remains a valid solution to the SOI problem under the atmospheric conditions at Maui. Table 5 shows the maximum P_{FA} results for each test.

From the reduced data set test iterations, there was found to be no statistically significant difference between classification accuracies at the different levels of turbulence. This implies that the hybrid adaptive optics system is not really needed for the FSTNN to achieve good results for the SOI problem. Furthermore, the variance realized by these runs implies that the standard FSTNN is the least sensitive to the level of atmospheric turbulence.

Turbulence	*Data Set 1*	*Data Set 2*
None	20.00	2.17
Low	32.86	6.52
Medium	22.86	6.52
High	18.57	10.87
Hybrid AO	51.43	13.04

Table 5. Maximum P_{FA} results (%) for the standard FSTNN algorithm.

Since data set 2 is defined by shorter time intervals and descending passes only, which can be easily implemented in actual SOI tests at AMOS, its classification results are what one should expect AMOS to be able to realize in reality.

5.1.2 FSTNN with DTW. Table 6 shows the results of the tests run on the FSTNN using DTW. These results are consistently better than those of the standard FSTNN.

Turbulence	*Data Set 1*	*Data Set 2*
None	14.29	2.17
Low	24.29	2.17
Medium	21.43	2.17
High	14.29	6.52
Hybrid AO	47.14	4.35

Table 6. Maximum P_{FA} results (%) for the DTW algorithm.

As with the standard FSTNN, no statistically significant difference in the classification accuracies exists at the various turbulence levels. While the DTW method is slightly more sensitive to turbulence (i.e. more variance at each level), it consistently outperforms the standard FSTNN for any given realization of turbulence.

5.1.3 FSTNN with UTW. When using all of the data points in sets 1 and 2, the results of the FSTNN with UTW are not consistently better than those of the standard FSTNN, as shown in Table 7.

The reduced data set runs show that, on the average, UTW outperforms both DTW and the standard FSTNN, but it is very sensitive to turbulence. For any given realization of the OTF, it may or may not work better. This is because the implementation of UTW induces

Turbulence	*Data Set 1*	*Data Set 2*
None	34.29	15.22
Low	34.29	15.22
Medium	42.86	8.70
High	20.00	6.52
Hybrid AO	51.43	4.35

Table 7. Maximum P_{FA} results (%) for the UTW algorithm.

an additional source of error because it requires the test set to be interpolated. How close the linear interpolation is to the "true" trajectory has more bearing on the UTW algorithm's performance than on that of the other two methods.

5.2 Implementation

The DTW and UTW FSTNN algorithms can be easily implemented into the existing MATLAB FSTNN code by replacing the "fst_tst.m" and "fst_trn.m" functions with either the "fst_dtw.m" or "fst_utw.m" functions provided in the Appendix. Each of these new functions perform both training and testing, so calling the "fst_trn.m" function prior to testing is not necessary.

5.3 Feature Saliency

A metric for gauging the saliency of each feature was derived and shown in a brief example to be useful in determining important features for the standard FSTNN and FSTNN with UTW. However, the dynamic nature of the DTW algorithm makes the saliency metric's legitimacy for this method questionable at best.

5.4 Recommendations for Future Research

This thesis has presented two methods for improving the classification accuracy of the FSTNN by reducing the error induced by improper synchronization of the test and training sets. The FSTNN may be further improved by reducing other sources of error.

5.4.1 Continuous Feature State Transitions. The amount of error induced by assuming a linear trajectory between the vertices is unknown. An analysis to determine when this assumption is good and when it is not, and what other fits may be appropriate, would be very useful.

5.4.2 Feature Extraction. A classifier is only as good as the features used. One would find it difficult to discriminate between ping-pong balls and bowling balls by comparing spherical eccentricity. However, weight would be an excellent feature to use in such a problem. In most classification problems, the choice of features is not that obvious. Brandstrom tested the standard FSTNN using various types of either block or wedge Fourier coefficients. However, combinations of various block and wedge coefficients were not tested at the same time. Using the feature saliency metric derived in this thesis, it may be possible to test a multitude of wedge, block, ring, and other coefficients for saliency in an attempt to find an optimal combination of features useful for discrimination.

5.4.3 Feature Saliency. While the saliency metric derived in this thesis was shown to be useful in a small example, this result is anecdotal. Time constraints prevented a detailed analysis of its effectiveness. Future analysis of this metric is necessary, as well as developing a more meaningful metric for the DTW FSTNN which can take into account the effect the features have on the algorithm's dynamic synchronization.

5.4.4 Trajectory Clustering. Running the tests can be very time consuming if the population of training sequences is large. It would be useful to derive a method for finding a single trajectory or small set of trajectories which adequately represent the population of training trajectories. Zeek attempted some methods for accomplishing this with marginal results [23]. The best method used was to find a few trajectories from the population which were of a minimum average distance to the other members in the population.

One way to improve upon these results may be the use of a conjugate gradient algorithm to iteratively adjust the vertices of the minimum-distance trajectory to improve accuracy. An

approach similar to this was used by Neiberg and Casasent to prune the number of vertices in the training trajectories [10]. The use of a genetic algorithm to derive an optimal trajectory may also prove useful.

5.4.5 Stretch Factors. Very little effort went into finding the stretch factors used in the DTW algorithm for the experiments in this thesis. In truth, they were a shot in the dark which happened to work well. It is possible, even probable, that these stretch factors are nowhere near optimal. Furthermore, the sensitivity of the DTW algorithm to the stretch factors for the SOI problem is not known. A more rigorous analysis of the stretch factors may very well yield even better results from the FSTNN with DTW.

Appendix A. Matlab Code

```
%FST_DTW      This program compares trajectories using the
%             Dynamic Time Warping algorithm.  This code
%             is a modified version of FST_TST written by
%             Gary Brandstrom.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The function file is                 ../matlab/fst_dtw.m
% usage                        [D,Pp,projind] = fst_dtw(T,P)
%
%%%%%%%%%%%%%%%%%%%%%%%%%  variables in %%%%%%%%%%%%%%%%%%%%%%
%
%  T =  a matrix where each row represents a feature vector
% from a known image sequence.
%
%  P =  a matrix where each row represents a feature vector
% from a unknown image sequence.
%
%%%%%%%%%%%%%%%%%%%%%%%%%  variables out  %%%%%%%%%%%%%%%%%%%%
%
%  D =  vector representing distance from corresponding points
% on the trajectories
%
%  Pp = uniformly spaced points on training trajectory
%       corresponding to points Tp on the test trajectory
%
%  projind = vector of indices corresponding to the trajectory
%            links that the points in P project to
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [D,Pp,projind] = fst_dtw (T,P)

[ss,dim]=size(P);
[s,dim]=size(T); % s = number of points (samples) in trajectory,
% dim = length of feature vector

Pptemp = zeros(s,dim);  %initialize matrix

% calculate total length of known (trained) trajectory
```

```
c = T*T';
for i=1:s-1,
  temp=T(i+1,:)-T(i,:);  % this is (x2-x1)
  v(i,:)=temp/sqrt(temp*temp');
  len(i)=sqrt(temp*temp'); % this is ||x2-x1||
end;

% calculate total length of test trajectory
for i=1:ss-1,
  temp2=P(i+1,:)-P(i,:);  % this is (x2-x1)
  len2(i)=sqrt(temp2*temp2'); % this is ||x2-x1||
end;

ind = 1;

for n=1:ss,   % 1:s loop once for each test point (P)
  u=T(1,:)-P(n,:); % distance of nth test point to first point of trajectory
  d(n,1)=sqrt(u*u');
  Pptemp(1,:)=T(1,:); %% Pptemp is P prime

%%% next find distance of nth test point to each segment
    ef=T*P'; %% e is (i,n) and f is (i+1,n)
 for i=1:s-1;  % for each segment
    u=P(n,:)-T(i,:);
    alpha=u*v(i,:)';

    if alpha<=0,
      temp=P(n,:)-T(i,:);
      d(n,i+1)=sqrt(temp*temp');
      Pptemp(i+1,:)=T(i,:);
    elseif alpha>len(i),
      temp=P(n,:)-T(i+1,:);   %% new temp is (P-x2)
      d(n,i+1)=sqrt(temp*temp');
      Pptemp(i+1,:)=T(i+1,:);
    else,
      a=1-alpha/len(i);
      b=alpha/len(i);
      d(n,i+1)=P(n,:)*P(n,:)' - 2*a*ef(i,n) - 2*b*ef(i+1,n) + ...
              2*a*b*c(i,i+1) + a*a*c(i,i) + b*b*c(i+1,i+1);
      d(n,i+1)=sqrt(d(n,i+1));
      Pptemp(i+1,:)=a*T(i,:)+b*T(i+1,:);
    end
  end

puredist = d;
```

```
if ind > 1 d(n,ind)=1.5*d(n,ind); end; %stretch factor across
if ind < s-1 d(n,ind+2)=2*d(n,ind+2); end;  %stretch factor up

endtraj = ind + 2;
if ind >= s-1 endtraj = s; end;

[D(n),ind2]=min(d(n,ind:endtraj));  % min distance from P to trajectory
ind = ind + ind2 - 1;
D(n) = puredist(n,ind);
projind(n)=ind;
Pp(n,:)=Pptemp(ind,:); % point on traj corresp to min dist
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%FST_UTW     This program compares trajectories using the
%            Uniform Time Warping algorithm.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The function file is                ../matlab/fst_tst.m
% usage                        [D,Pp,Tp] = fst_utw(T,P,bz)
%
%%%%%%%%%%%%%%%%%%%%%%%  variables in %%%%%%%%%%%%%%%%%%%%%%
%
%  T =  a matrix where each row represents a feature vector
% from a known image sequence.
%
%  P =  a matrix where each row represents a feature vector
% from a unknown image sequence.
%
%  bz  = segmentation factor
%
%%%%%%%%%%%%%%%%%%%%%%%  variables out  %%%%%%%%%%%%%%%%%%%%
%
%  D =  vector representing distance from each point of P to
% the trajectory
%
%  Pp = projection points on training trajectory
%
%  Tp = projection points on test trajectory
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [D,Pp,Tp] = fst_utw(T,P,bz)

[ss,dim]=size(P);
[s,dim]=size(T);

% calculate total length of known (trained) trajectory
c = T*T';
for i=1:s-1,
  temp=T(i+1,:)-T(i,:);  % this is (x2-x1)
  v(i,:)=temp/sqrt(temp*temp');
  len(i)=sqrt(temp*temp'); % this is ||x2-x1||
end;
trainlen = sum(len);

% calculate total length of test trajectory
c2 = P*P';
for i=1:ss-1,
  temp2=P(i+1,:)-P(i,:);  % this is (x2-x1)
  v2(i,:)=temp2/sqrt(temp2*temp2');
  len2(i)=sqrt(temp2*temp2'); % this is ||x2-x1||
end;
testlen = sum(len2);


% calculate bz*s+1 evenly spaced points on the training trajectory
Pp(1,:) = T(1,:);
Pp(bz*s+1,:) = T(s,:);

for i=1:bz*s-1
   j=1;
   tlen=0;
   while j<s
      tlen=tlen+len(j);
      if (tlen)>=(i*trainlen/(bz*s))
         Pp(i+1,:) = T(j,:) + ((i*trainlen/(bz*s))-(tlen-len(j)))*v(j,:);
         j = s-1;
      end;
      j = j+1;
   end;
end;

% calculate bz*s+1 evenly spaced points on the test trajectory
Tp(1,:) = P(1,:);
Tp(bz*s+1,:) = P(ss,:);
```

```
for i=1:bz*s-1
   j=1;
   tlen = 0;
   while j<ss
      tlen=tlen+len2(j);
      if (tlen)>=(i*testlen/(bz*s))
         Tp(i+1,:) = P(j,:) + ((i*testlen/(bz*s))-(tlen-len2(j)))*v2(j,:);
         j = ss-1;
      end;
      j = j+1;
   end;
end;

% Calculate distances between corresponding points on the trajectories
for n=1:(bz*s)
   tempdist = Pp(n,:) - Tp(n,:);
   D(n) = sqrt(tempdist*tempdist');
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Bibliography

1. Applied Optics Group, Imperial College Homepage. http://op.ph.ic.ac.uk.

2. Barmore, Gary D. *Speech Recognition Using Neural Nets and Dynamic Time Warping.* Masters Thesis, Air Force Institute of Technology, December 1988.

3. Bate, Roger R, et al. *Fundamentals of Astrodynamics.* Dover Publications, Inc., 1971.

4. Brandstrom, Gary W. *Space Object Identification Using Spatio-Temporal Pattern Recognition.* Masters Thesis, Air Force Institute of Technology, December 1995.

5. Cover, T.M. and P.E. Hart. "Nearest Neighbor Pattern Classification," *IEEE Transactions on Information Theory, 13*:21–27 (1967).

6. Gary W. Brandstrom, et al. "Space Object Identification Using Spatio-Temporal Pattern Recognition," *Proc. Soc. Photo-Opt. Instrum. Eng. (SPIE), 2760*:475–486 (1996).

7. Gregg, Dan W. *Decision Boundary Analysis of Feature Selection for Breast Cancer Diagnosis.* Masters Thesis, Air Force Institute of Technology, March 1997.

8. Lee, Chulhee and David Landgrebe. "Feature Extraction Based on Decision Boundaries," *IEEE Transactions on Pattern Analysis and Machine Intelligence, 15, no.4*:388–400 (1993).

9. Neiberg, Leonard and David. P. Casasent. "Feature space trajectory (FST) classifier neural network," *Proc. Soc. Photo-Opt. Instrum. Eng. (SPIE), 2353*:276–292 (1994).

10. Neiberg, Leonard and David P. Casasent. "Classifier and shift-invariant automatic target recognition neural networks," *Neural Networks, 8 no.7/8*:1117–1129 (1995).

11. Neiberg, Leonard and David P. Casasent. "Feature space trajectory neural net classifier: 8-class distortion invariant tests," *Proc. Soc. Photo-Opt. Instrum. Eng. (SPIE), 2588*:540–555 (1995).

12. Neiberg, Leonard and David P. Casasent. "Feature space trajectory neural network classifier," *Proc. Soc. Photo-Opt. Instrum. Eng. (SPIE), 2492*:361–372 (1995).

13. Neiberg, Leonard and David P. Casasent. "Feature space trajectory neural network classifier: confidences and thresholds for clutter and low contrast objects," *Proc. Soc. Photo-Opt. Instrum. Eng. (SPIE), 2760*:435–446 (1996).

14. Neiberg, Leonard and David P. Casasent. "Time-sequential neural net classifier and pose estimator," *World Congress on Neural Networks*, 345–350 (1996).

15. Ney, Hermann. "The Use of a One-Stage Dymanic Programming Algorithm for Connected Word Recognition," *IEEE Trans. on Speech and Audio Processing, ASSP-32, no.2*:263–271 (1984).

16. Rogers, Steven K. and Matthew Kabrisky. *An Introduction to Biological and Artificial Neural Networks for Pattern Recognition.* SPIE Press, 1991.

17. Roggemann, Michael C. "Documentation for Simulation Programs: OTFSIM3 and HYSIM3." from Air Force Institute of Technology, 1996.

18. Roggemann, Michael C. and Byron Welsh. *Imaging Through Turbulence.* CRC Press, 1996.

19. Ruck, Dennis W. "Feature Selection Using a Multilayer Perceptron," *Journal of Neural Network Computing, 2, no.2* (1990).

20. Stewart, James A. *Nonlinear Time Series Analysis.* Masters Thesis, Air Force Institute of Technology, March 1994.

21. Suzuki, A. "Maui AMOS brief." from Phillips Lab, 1994.

22. Wood, Gregory E. *Estimation of Satellite Orientation from Space Surveillance Imagery Measured with an Adaptive Optics Telescope.* Masters Thesis, Air Force Institute of Technology, December 1996.

23. Zeek, Eric J. *Speaker Recognition by Hidden Markov Models and Neural Networks.* Masters Thesis, Air Force Institute of Technology, December 1996.

Vita

Neal W. Bruegger [redacted] He earned a B.S. in Aerospace Engineering from the University of Colorado at Boulder in 1992. Upon completion of this degree, he entered the US Air Force as a communications officer. His first duty assignment was as a Base Network Control Center OIC at the 72d Communications Squadron, Tinker AFB, Oklahoma. Upon completion of his M.S. in Operations Research at the Air Force Institute of Technology, he will work as an operations research analyst at Headquarters Air Force Operational Test and Evaluation Center at Kirtland AFB, New Mexico.

[redacted]

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden. to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
March 1997

3. REPORT TYPE AND DATES COVERED
Master's Thesis

4. TITLE AND SUBTITLE
Space Object Identification Using Feature Space Trajectory Neural Networks

5. FUNDING NUMBERS

6. AUTHOR(S)
Capt Neal W. Bruegger

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GOR/ENG/97M-02

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Phillips Lab/OL-YY
535 Lippoa Pkwy, Ste 200
Kihei, Maui, HI 96753

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public release; Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
The Feature Space Trajectory Neural Network (FSTNN) is a simple yet powerful pattern recognition tool developed by Neiberg and Casasent for use in an Automatic Target Recognition System. Since the FSTNN was developed, it has been used on various problems including speaker identification and space object identification. However, in these types of problems, the test set represents time-series data rather than an independent set of points. Since the distance metric of the standard FSTNN treats each test point independently without regard to its position in the sequence, the FSTNN can yield less than optimal results in these problems. Two methods for incorporating sequence information into the FSTNN algorithm are presented. These methods, Dynamic Time Warping (DTW) and Uniform Time Warping (UTW), are described and compared to the standard FSTNN performance on the space object identification problem. Both reduce error induced by improper synchronization of the test and training sequences and make the FSTNN more generally applicable to a wide variety of pattern recognition problems. They incorporate sequencing information by synchronizing the test and training trajectories. DTW accomplishes this "on-the-fly" as the sequence progresses while UTW uniformly compensates for temporal differences across the trajectories. These algorithms improve the maximum probability of false alarm (P_{FA}) of the standard FSTNN by an average of 10.18% and 27.69%, respectively, although UTW is less consistent in its results. A metric for determining the saliency of the features in an FSTNN is also presented and demonstrated.

14. SUBJECT TERMS
Pattern Recognition, Space Object Identification, Feature Extraction, Feature Saliency, Spatio-Temporal Analysis, Feature Space Trajectory Neural Net, Neural Nets

15. NUMBER OF PAGES
78

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL